Osborne/McGraw-Hill

CP/M

MANUAL PARA PROGRAMADORES

Andy Johnson-Laird

LIBRERIA
Follas Novas
Montero Ríos, 37
Teléfonos 59 44 06
59 44 18
SANTIAGO

CP/M MANUAL PARA PROGRAMADORES

CONSULTORES EDITORIALES
AREA DE INFORMATICA Y COMPUTACION

Antonio Vaquero Sánchez
Catedrático de Informática
Facultad de Ciencias Físicas
Universidad Complutense de Madrid
ESPAÑA

María Lourdes Fournier García
Actuaria, Facultad Ciencias UNAM
Profesora Asociada a Tiempo Completo
Universidad Autónoma Metropolitana
MEXICO

Gerardo Quiroz Vieyra
Ingeniero en Comunicaciones y Electrónica
Escuela Superior de Ingeniería Mecánica y Eléctrica IPN
Carter Wallace, S. A.
Universidad Autónoma Metropolitana
Docente DCSA
MEXICO

Alfonso Pérez Gama
Ingeniero Electrónico
Universidad Nacional de Colombia
COLOMBIA

José Portillo
Universidad de Lima
PERU

Elías Lopata Szmiga
Departamento de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad Metropolitana —UNIMET—
VENEZUELA

CP/M MANUAL PARA PROGRAMADORES

Andy Johnson-Laird

Traducción:

Jordi Aguiló Llobet
Director Departamento de Informática
Facultad de Ciencias
Universidad Autónoma de Barcelona

Revisión técnica:

Antonio Vaquero Sánchez
Catedrático de Informática
Facultad de Ciencias Físicas
Universidad Complutense de Madrid

OSBORNE/McGraw-Hill

MADRID • BOGOTA • BUENOS AIRES • GUATEMALA • LISBOA • MEXICO
NUEVA YORK • PANAMA • SAN JUAN • SANTIAGO • SAO PAULO
AUCKLAND • HAMBURGO • JOHANNESBURGO • LONDRES • MONTREAL • NUEVA DELHI
PARIS • SAN FRANCISCO • SINGAPUR • ST. LOUIS • SIDNEY • TOKIO • TORONTO

CP/M MANUAL PARA PROGRAMADORES

Prohibida la reproducción total o parcial de esta obra
por cualquier medio, sin autorización escrita del editor.

DERECHOS RESERVADOS © 1986 respecto a la primera edición en español por
LIBROS McGRAW-HILL DE MEXICO, S. A. DE C. V.
Atlacomulco 499-501, Naucalpan de Juárez, Edo. de México
Miembro de la Cámara Nacional de la Industria Editorial, Reg. núm. 465

ISBN: 968-451-756-4

Traducido de la primera edición en inglés de
THE PROGRAMMER'S CP/M HANDBOOK

Copyright © 1983, by McGraw-Hill, Inc., U.S.A.

ISBN: 0-88134-103-7

Edición exclusiva para Ediciones La Colina, S. A. (España)

ISBN: 84-7615-083-0
Depósito legal: M. 39098-1985

Compuesto en Fernández Ciudad, S. L.
Impresión: Gráficas EMA. Miguel Yuste, 27. 28037 Madrid

PRINTED IN SPAIN - IMPRESO EN ESPAÑA

Dedicatoria

Hace muchos años que escuché que "Perfección consistía en una educación británica, un salario americano y una esposa japonesa".

De acuerdo con ello, quiero dar las gracias a los miembros de Culford School, que me han proporcionado una educación británica, a mis compañeros de trabajo en Johnson-Laird Inc. y en Control-C Software y a sus clientes, que me han proporcionado un salario americano, y al señor y a la señora Kitagawa, que me han dado a Kay Kitagawa (quien no sólo es mi esposa, sino que además corrige mi gramática inglesa).

A.J.-L.

Agradecimientos

Aunque este libro no ha sido autorizado ni sancionado por Digital Research, quisiera expresar mi agradecimiento a Gary Kildall y Kathy Strutynski, de Digital Research, y a Phil Nelson (anteriormente de Digital Research, actualmente de Victor Technology) por su ayuda para que me mantuviese en el camino de la verdad en este libro. Quisiera dar las gracias también a Denise Penrose, Marty McNiff, Mary Borchers y Ralph Baumgartner, en Osborne/McGraw-Hill, por su al parecer inextinguible paciencia.

A. J.-L.

Contenido

1 Introducción 1

2 La estructura del CP/M 7

3 El sistema de ficheros del CP/M 21

4 El procesador de órdenes de consola (CCP) 53

5 El sistema operativo básico de disco 77

6 El sistema básico de entrada/salida 161

7 Construcción de un nuevo sistema CP/M 199

8 Escritura de un BIOS mejorado 225

9 Procesamiento de errores de hardware 317

10 Comprobación detenida de un nuevo BIOS 341

11 Programas de utilidad adicionales 399

12 Mensajes de error 481

A Conjunto de caracteres ASCII 501

B Resumen de órdenes CP/M 505

C Resumen de las llamadas del BDOS 517

D Resumen de las llamadas del BIOS 523

Indice 527

Descripción del contenido
Notación
Ejemplos de programas en disquete

1

Introducción

Este libro es una continuación a la *Guía del usuario del CP/M Osborne* de Thom Hogan. Es un libro técnico escrito principalmente para programadores que necesitan un conocimiento profundo de la estructura interna del CP/M —cómo trabajan las distintas piezas del CP/M, cómo utilizar CP/M como sistema operativo y, finalmente, cómo implementar el CP/M en diferentes computadoras—. Este libro está dedicado a aquellos que

- han trabajado con microcomputadoras que funcionan con el sistema operativo CP/M de Digital Research;
- comprenden las interioridades del mundo de los microprocesadores —bits, bytes, puertos, RAM, ROM y toda la jerga de los programadores;
- saben escribir en lenguaje ensamblador para los chips, unidad central de procesamiento, Intel 8080 o Zilog Z80.

Si el lector no posee este tipo de conocimientos básicos, deberá comenzar adquiriendo experiencia práctica en un sistema que funcione con CP/M y leyendo los siguientes libros de Osborne/McGraw-Hill:

- *Introducción a los microcomputadores: Volumen I. Conceptos básicos*
  Este libro describe los conceptos fundamentales y los aspectos de los mismos que es necesario conocer acerca de los microprocesadores para programarlos. Si se carece realmente de conocimientos elementales, puede utilizarse el volumen 0 titulado *El libro del principiante*.
- *Programación en el lenguaje ensamblador 8080A/8085*
  Este libro cubre todos los aspectos de la escritura de programas en lenguaje ensamblador del 8080, presentando múltiples ejemplos.
- *Guía del usuario de CP/M de Osborne*
  Este libro introduce el sistema operativo CP/M. Enseña a utilizar éste como herramienta para obtener datos determinados en una computadora.

El libro que ahora presentamos se ocupa únicamente de la versión 2.2 de CP/M para los chips 8080 o Z80. Durante su período de redacción han aparecido nuevas versiones de CP/M y de MP/M (el sucesor multiusuario y multitarea de CP/M). Han aparecido el CP/M-86 y el MP/M-86 para el chip Intel 8086 y el MP/M-II para los chips 8080 o Z80, con CP/M 3.0 (8080 o Z80) en los talones. El 8086, aunque relacionado estructuralmente con el 8080, difiere de éste lo suficiente como para hacer imposible el detallarlo en este libro; y aunque MP/M-II y MP/M-86 son similares a CP/M, presentan múltiples aspectos que no se adaptan a la finalidad de este libro.

Descripción del contenido

Este libro aborda los temas desde el vértice superior de la pirámide, aportando sucesivas capas sobre la misma relacionadas con el mismo material, pero proporcionando más detalles.

El primer capítulo incluye una breve descripción de la notación utilizada en este libro para los programas ejemplo escritos en lenguaje ensamblador de Intel 8080 y en el lenguaje de programación C.

El capítulo 2 está dedicado a la estructura del CP/M, describiendo sus partes principales, sus posiciones en memoria y sus funciones.

El capítulo 3 se ocupa del sistema de ficheros del CP/M, con el máximo detalle posible, dada su naturaleza. Se describen la entrada al directorio, el bloque de parámetros del disco y la organización de ficheros.

El capítulo 4 está dedicado al procesador de órdenes de consola —Console Commands Processor (CCP)—, examinando la forma de introducir líneas de órdenes, las órdenes de CP/M construidas en el CCP, la forma en que el CCP carga los programas y cómo transfiere el control a estos programas.

En el capítulo 5 comienza la sección de programación. Maneja las llamadas del sistema que los programas pueden hacer a la parte de alto nivel de CP/M, el sistema operativo básico de disco (Basic Disk Operating System, BDOS).

Los capítulos 6 al 10 se refieren al sistema básico de entrada/salida (BIOS). Esta es la parte de CP/M que es distinta en cada computadora. Es la parte que el programador escribirá e implementará para su propia computadora.

El capítulo 6 describe una implementación estándar del BIOS.

El capítulo 7 describe el mecanismo de reinstalación de CP/M para una configuración distinta.

El capítulo 8 muestra la forma de escribir un BIOS mejorado.

El capítulo 9 contempla detenidamente la forma de manejar los errores de hardware —cómo detectarlos y corregirlos, haciendo esta tarea más sencilla para la persona que utiliza la computadora.

El capítulo 10 aborda los problemas que pueden presentarse cuando se desarrolla un código BIOS propio. Incluye subrutinas para dicho desarrollo y describe técnicas para ahorrar tiempo y preocupaciones.

El capítulo 11 describe programas de utilidad, algunos de los cuales funcionan con las características del BIOS mejorado del capítulo 8 y otros que pueden trabajar con todas las implementaciones del CP/M 2.

El capítulo 12 se refiere a mensajes de error y a algunas particularidades que se descubrirán, a veces con grandes esfuerzos, en el CP/M. Se explican los mensajes y se documentan algunas de las probables causas que provocan resultados extraños.

Los apéndices contienen información práctica, así como resúmenes de informaciones que pueden necesitarse para diseñar, codificar y comprobar programas que funcionen en CP/M o en sus rutinas BIOS propias.

Notación

Al programar una computadora, el programador estará sentado ante el terminal dialogando con CP/M y con los programas de utilidad que funcionan supeditados a éste. Las secciones que siguen describen la notación utilizada para representar el diálogo que aparece en el terminal y la salida que aparece en la impresora.

Diálogo de consola

Este libro sigue los convenios utilizados en la *Guía del usuario del CP/M Osborne,* ampliados ligeramente para manejar diálogos más complejos. En este libro

- <nombre> significa el carácter ASCII nombrado entre los corchetes angulares, < y >. Por ejemplo, <BEL> es el carácter ASCII timbre o campana (bell) y <HT> es el carácter ASCII de tabulación horizontal (Horizontal Tab). (Véase el apéndice A para el conjunto completo de caracteres ASCII.)
- <cr> significa presionar (pulsar) la tecla de RETORNO DE CARRO (CARRIAGE RETURN).
- 123 o un número sin sufijo significa un número decimal.
- 100B o cualquier número seguido de B significa un número binario.
- 0A5H o un número cualquiera seguido de H significa un número hexadecimal. Un número hexadecimal que comienza por una letra irá normalmente precedido de un 0 para evitar confusiones.
- ^x significa mantener oprimida la tecla de control (CTRL) mientras se pulsa la tecla x.
- Los subrayados indican entradas escritas sobre el teclado. La salida de la computadora se muestra sin subrayar.

Ejemplos de programas en lenguaje ensamblador

Este libro utiliza los nemotécnicos del Intel 8080 desde el principio hasta el fin como un "mínimo común denominador" —la CPU Z80 contiene particularidades de las que carece el 8080, pero no a la inversa—. La salida del ensamblador ASM de Digital Research se muestra de tal forma que pueda verse el código objeto generado, así como el fuente.

Ejemplos en lenguaje de alto nivel

Los programas de utilidad descritos en el capítulo 11 están escritos en C, un lenguaje de programación que se presta especialmente a la descripción clara de algoritmos sin perderse en burocracias lingüísticas. Se han evitado expresiones rebuscadas en favor de aquellas que muestran de la forma más clara posible cómo resolver el problema. Se comenta ampliamente el código.

Un libro excelente para aquellos que no saben programar en C es *El lenguaje C de programación* de Brian Kernighan y Dennis Ritchie (Prentice-Hall). El apéndice A de este libro es el manual de referencia C.

Ejemplos de programas en disquete

Los ejemplos de programas de este libro han sido ensamblados con ASM y comprobados con DDT (Digital Research's Dynamic Debugging

Tool), la herramienta de depuración dinámica de Digital Research. Los ejemplos en C se compilaron utilizando el compilador C BDS de Leor Zolman (versión 1.50) y comprobados utilizando el BIOS mejorado del capítulo 8.

Todo lo que se presenta en este libro en código fuente está disponible en un disquete de 8 pulgadas de una sola cara y simple densidad (formato IBM 3740). Para solicitar este disquete, pueden llamar o escribir a

Johnson-Laird, Inc.
Attn: The CP/M Programmer's Handbook Diskette
6441 SW Canyon Court
Portland, OR 97221
Tel: (503) 292-6330

El disquete se vende al precio de 50 dólares más gastos de envío.

El CP/M de Digital Research
Las piezas del CP/M
- Formato del disquete CP/M
- Carga del CP/M
- Procesador de órdenes de consola (CCP)
- Sistema operativo básico de disco (BDOS)
- Sistema básico de entrada/salida (BIOS)
- Interacciones del CCP, BDOS y BIOS

2

La estructura del CP/M

Este capítulo introduce las piezas que componen CP/M —cómo son y cómo trabajan—. Esta visión general del CP/M establecerá un armazón previo a los capítulos posteriores, que añadirán información más detallada.

Puede haberse adquirido la versión estándar de CP/M directamente de Digital Research, pero probablemente se ha adquirido el CP/M al comprar el sistema microprocesador o su sistema de discos. O puede haberse adquirido CP/M sólo de un distribuidor de software. En cualquier caso, este distribuidor o la compañía que fabrica el sistema o el disco ya habrá modificado la versión estándar de CP/M adaptándola para que trabaje en su hardware particular. La mayor parte de las versiones de CP/M que se encuentran en el mercado tienen más ficheros en su sistema de disquetes que los que se describen aquí como versión estándar de Digital Research.

Algunos fabricantes han reescrito toda la documentación, por lo que el comprador no recibe todos los manuales CP/M de Digital Research. Si este es el caso, habrán de solicitarse estos manuales a Digital Research, porque el programador los necesita como referencia.

El CP/M de Digital Research

Digital Research proporciona una versión estándar del CP/M "con sabor a vainilla", que funciona sólo en el sistema de desarrollo de las microcomputadoras Intel (MDS). El paquete de programas CP/M de Digital Research contiene siete manuales y un disquete estándar de 8 pulgadas, de una sola cara y simple densidad, formato IBM 3740.

Con este sistema CP/M van los siguientes manuales:

- *Una introducción a las particularidades y habilidades del CP/M.* Se trata de una breve descripción de CP/M y de los programas de utilidad que se encuentran en el disquete. Describe únicamente la versión 1.4 de CP/M.
- *Guía del usuario del CP/M 2.0.* Digital Research escribió este manual para describir las nuevas características del CP/M 2.0 y las extensiones realizadas al CP/M 1.4.
- *ED: Un editor de texto para el sistema CP/M de disco.* Frente a los productos estándar actuales, ED es un editor de líneas primitivo, pero puede utilizarse también para realizar cambios en los ficheros que contienen texto ASCII, tales como el código de fuente del BIOS.
- *Ensamblador del CP/M (ASM).* ASM es un ensamblador sencillo pero rápido que puede utilizarse para traducir el código fuente BIOS del disquete al código máquina. Como quiera que el ASM es simplemente un ensamblador en esqueleto, muchos programadores utilizan actualmente su sucesor, el MAC (también de Digital Research).
- *La herramienta de depuración dinámica del CP/M —Dinamic Debugging Tool (DDT)—.* El DDT es un programa extremadamente útil que permite cargar programas en formato de código máquina y comprobarlos a continuación, ejecutando el programa bien una instrucción máquina cada vez o bien deteniéndose sólo cuando la CPU alcanza un punto específico en el programa.
- *Guía de alteraciones del CP/M.* Existen dos manuales con este título, uno de la versión 1.4 del CP/M y otro para la 2.0. Ambos manuales describen, en forma codificada, cómo modificar el CP/M.
- *Guía de interfases del CP/M.* Existen también dos versiones, para el 1.4 y el 2.0. Estos manuales indican cómo escribir programas que comuniquen directamente con el CP/M.

El disquete que suministra Digital Research posee los ficheros siguientes:

ASM.COM
El ensamblador del CP/M.

BIOS.ASM
Un fichero en código fuente que contiene una muestra de BIOS para el sistema de desarrollo de micros Intel (MDS). A menos que se posea el MDS, este fichero sólo es útil como ejemplo de BIOS.

CBIOS.ASM
Otro fichero en código fuente para BIOS. Este es tipo armazón: existen espacios de forma que se puedan insertar códigos para cada computadora particular.

DDT.COM
El programa herramienta de depuración dinámica.

DEBLOCK.ASM
Un fichero en código fuente que puede tenerse necesidad de utilizar en el BIOS en el caso de que la computadora utilice tamaños de sector distintos de 128 bytes. Es un ejemplo de cómo bloquear y desbloquear sectores de 128 bytes al tamaño de sector deseado o viceversa.

DISKDEF.LIB
Una biblioteca de textos fuente a utilizar cuando se tiene una copia del ensamblador avanzado MAC, de Digital Research.

DUMP.ASM
La fuente para un programa ejemplo. El DUMP lee un fichero CP/M en disco y lo presenta en forma hexadecimal en la consola.

DUMP.COM
El programa práctico ejecutable derivado del DUMP.ASM.

ED.COM
El editor de ficheros fuente.

LOAD.COM
Un programa que toma el fichero de salida del ensamblador, ASM, en código máquina y crea otro fichero con los datos reordenados de tal forma que se pueda ejecutar el programa únicamente mecanografiando su nombre en el teclado.

MOVCPM.COM
Un programa que crea versiones de CP/M para diferentes tamaños de memoria.

PIP.COM
Un programa para copiar información de un lugar a otro (PIP es la abreviatura de Peripheral Interchange Program).

STAT.COM
Un programa que muestra estadísticas sobre el CP/M y otras informaciones que se hayan almacenado en discos.

SUBMIT.COM
Un programa que se utiliza para entrar automáticamente órdenes CP/M. Ayuda a evitar el mecanografiado de secuencias de órdenes largas.

SYSGEN.COM
Un programa que escribe CP/M en disquetes.

XSUB.COM
Una versión extendida del programa SUBMIT. Los ficheros mencionados previamente se dividen en dos grupos: uno se utiliza únicamente para reconstruir el CP/M, mientras que el otro conjunto está formado por herramientas de programación de propósito general.

Las piezas del CP/M

El CP/M está formado por el sistema operativo básico de disco (BDOS), el procesador de órdenes de consola (CCP) y el sistema básico de entrada/salida (BIOS).

Ocasionalmente se presentan en los manuales del CP/M referencias a algo denominado FDOS, que significa "sistema operativo de disco flexible". Se da este nombre a la porción del CP/M formada por el BDOS y el BIOS y es una reliquia que ha permanecido desde la versión original. Como quiera que casi nunca es necesario referirse al BDOS y al BIOS combinados como una entidad única, no se aludirá más en este libro al FDOS.

El BDOS y el CCP son las partes características de CP/M. A menos que se desee pagar varios miles de dólares, no se puede obtener el código fuente de las mismas. Tampoco es necesario. El CP/M está diseñado de tal forma que todo lo referente al código que varía de una máquina a otra está contenido en el BIOS, y el programa de fuente del BIOS se obtiene en Digital Research. Algunas compañías construyen BIOS's especializados para diferentes computadoras. En muchos casos, al igual que algunos fabricantes de hardware CP/M, no está disponible el código fuente para sus BIOS; han empleado tiempo y trabajo en la construcción de estos BIOS, y desean preservar la naturaleza peculiar que le han dado.

A veces hay que construir una configuración especial del CP/M para una computadora determinada. Para ello no son necesarios más que los cuatro pasos siguientes:

1. Obtener una versión del BDOS y del CCP para el tamaño de memoria de la computadora que se utilice.

2. Escribir una versión modificada del BIOS, de acuerdo con el hardware de la computadora en cuestión.
3. Escribir un pequeño programa para cargar el CP/M en la memoria al presionar el botón de RESET de la computadora.
4. Ensamblar todas las piezas juntas y escribirlas en el disquete.

Estos pasos se explican en los capítulos 7, 8 y 9.

En el tercer paso se escribe un pequeño programa que carga el CP/M en la memoria al presionar el botón RESET de la computadora. Este programa se denomina normalmente cargador "de cordón de bota". También se llama a veces la "bota" o bien el cargador de arranque en frío. La denominación de "cordón de bota" surge de la idea de que al conectar una computadora por primera vez no hay ningún programa a realizar. La tarea de obtener este primer programa en la computadora es tan complicada conceptualmente como la de intentar levantarse a sí mismo del suelo tirando de los cordones de las propias botas. En los primeros tiempos de las computadoras esta operación se realizaba mediante la introducción manual de instrucciones posicionando gran cantidad de conmutadores (la computadora que estaba construida para leer los conmutadores apenas se conectaba). Hoy en día, las microcomputadoras contienen algunos fragmentos pequeños de programa en memoria "no volátil" de sólo lectura —memoria que retiene datos aun cuando la computadora se desconecta—. Este programa almacenado, normalmente como un chip de memoria de sólo lectura programable (PROM), puede cargar el programa de arranque en frío, que a su vez carga el CP/M.

Formato del disquete CP/M

La versión estándar de CP/M está dispuesta en un disquete de 8 pulgadas, de una sola cara. Los disquetes de otros tipos probablemente tienen distintas disposiciones; los discos duros son totalmente diferentes.

El formato físico del disquete estándar de 8 pulgadas se muestra en la figura 2-1. El disquete tiene un total de 77 pistas concéntricas numeradas de 0 (la más exterior) a 76 (la más interna). Cada una de estas pistas está dividida radialmente en 26 sectores. Estos sectores físicos están numerados del 1 al 26; el sector físico cero no existe. Cada sector tiene espacio suficiente para 128 bytes de datos.

Incluso cuando el CP/M se implementa en un disco duro mayor con tamaños de sector mucho mayores, trabajar con sectores de 128 bytes. El BIOS posee instrucciones especiales que convierten los sectores *reales* en sectores de 128 bytes de estilo CP/M.

Una nota final acerca del formato físico: el disquete de 8 pulgadas,

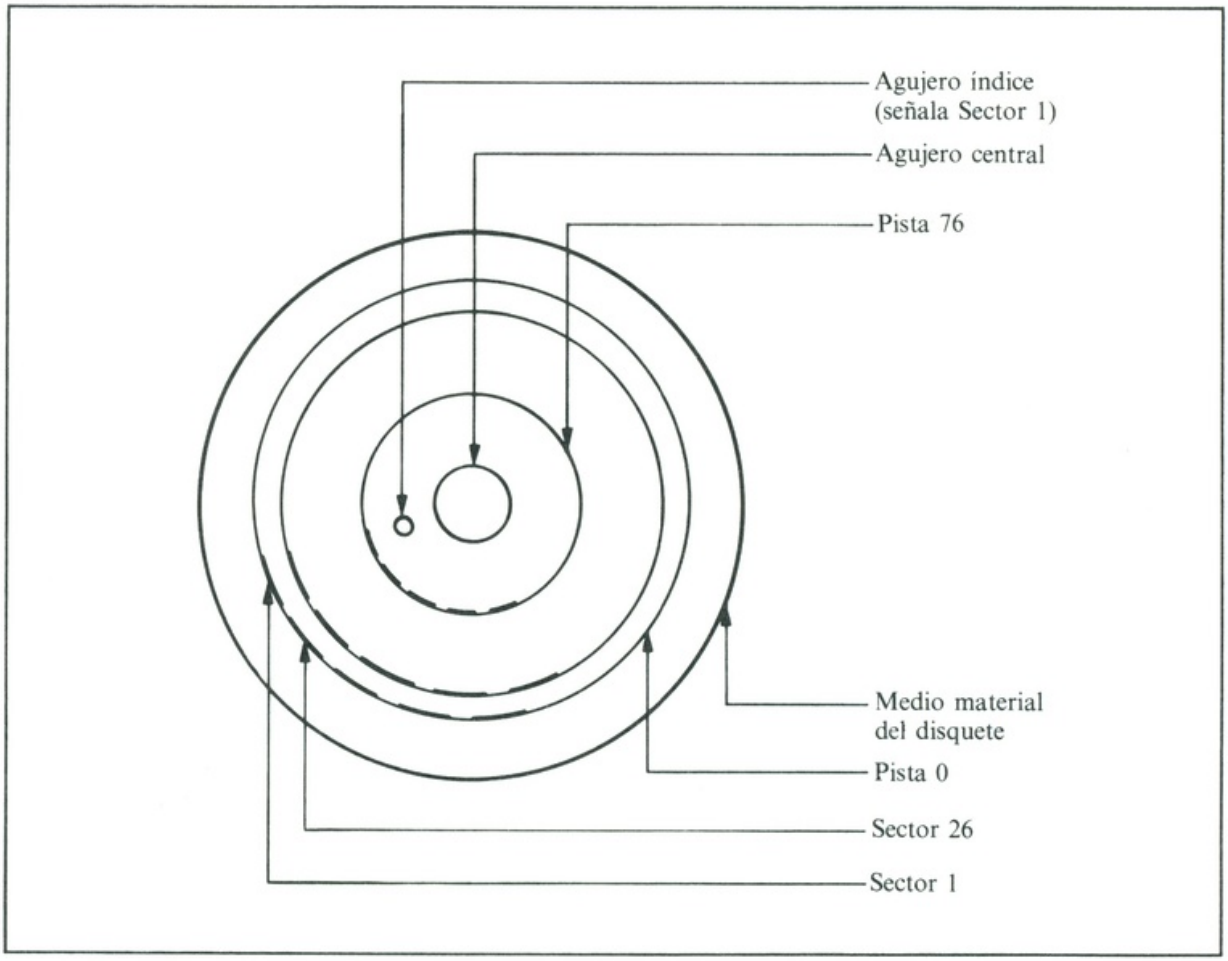


Figura 2-1. Disposición del disquete.

sectorizado por software, de una sola cara, simple densidad (formato IBM 3740) es el *único* formato estándar. Todos los formatos restantes serán exclusivos de los fabricantes de hardware que los usen. Es impensable que se pueda leer un disquete en una computadora de un fabricante si ha sido escrito en otra, aunque su forma sea aparentemente la misma. Por ejemplo, disquete de una sola cara, de doble densidad, escrito en un sistema de desarrollo Intel no puede ser leído en una computadora de Digital Microsystems, aun cuando ambas usen formato de doble densidad. Si se desea cambiar datos de una computadora a otra, hay que usar disquetes de 8 pulgadas, de una sola cara, con formato de simple densidad y entonces *funcionará*.

Para ver cómo está almacenado el CP/M en un disquete, obsérvense las dos primeras pistas del mismo, que son la 0 y la 1. La figura 2-2 muestra cómo se almacenan los datos en estas pistas.

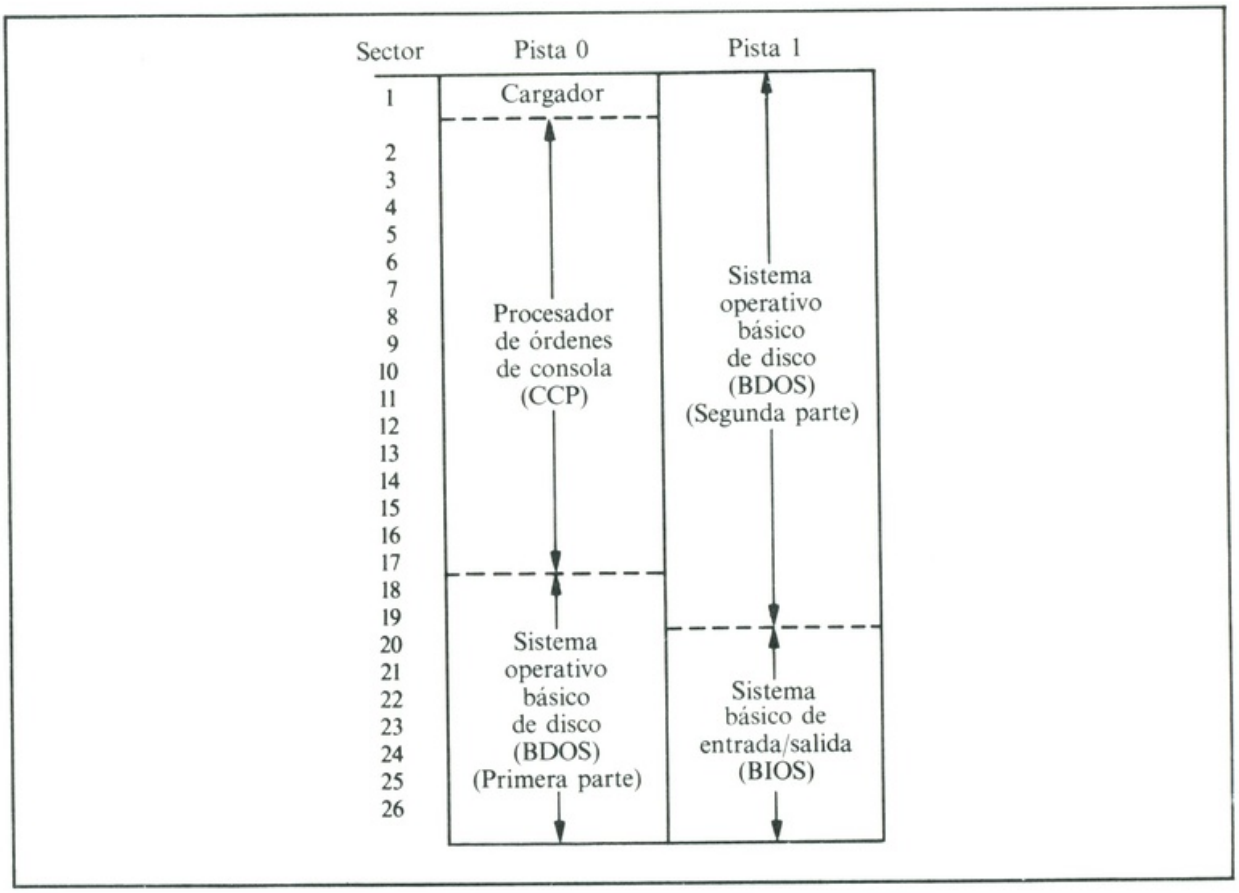


Figura 2-2. Disposición del CP/M en las pistas 0 y 1 del disquete.

Carga del CP/M

Los sucesos que ocurren después de conectar por primera vez la computadora y después de colocar el disquete en su unidad de disco son los mismos que los que ocurren al presionar el botón de RESET —la computadora genera una señal de RESET.

El botón de RESET detiene a la unidad central de procesamiento (CPU). Todas las variables internas de la CPU se colocan en el estado inicial y todos los registros se ponen a cero. El contador de programa se pone también a cero de forma que cuando la señal de RESET desaparece (permanece únicamente durante algunas milésimas de segundo), la CPU comienza a ejecutar instrucciones en el lugar 0000H de la memoria.

Los chips de memoria, al recibir por primera vez energía, no contienen ningún valor particular. Por consiguiente, los diseñadores de hardware hacen que algunas instrucciones iniciales sean forzadas en la memoria en el lugar 0000H y así sucesivamente. Aquí es donde parece que uno quiera

elevarse tirando de los cordones de las propias botas. ¿Cómo puede conseguirse que la computadora obedezca una instrucción determinada cuando no existe "nada" (con valor sensible) dentro de la máquina?

Existen dos técnicas comunes para colocar las instrucciones en la memoria:

De alimentación forzada (Force-feeding)

En esta opción el diseñador de hardware supone que cuando se aplica la señal de RESET alguna parte del sistema de cálculo, típicamente el controlador de disco, puede enmascararse como memoria. Exactamente antes de que el CPU se desconecte, el controlador del disco asumirá el control de la computadora y copiará un pequeño programa en la memoria en el lugar 0000H y sucesivos. A partir de este momento se permite a la CPU empezar la ejecución de instrucciones en la posición 0000H. El controlador del disco preserva las instrucciones incluso en caso de desconexión, porque éstas están almacenadas en firmware no volátil basado en PROM. Estas instrucciones hacen que el controlador del disco lea el primer sector de la primera lista del disquete del sistema en la memoria y entonces transfiere a ésta el control.

ROM fantasma (Shadow ROM)

Es una variación de la técnica de alimentación forzada. El fabricante del hardware dispone alguna ROM en el lugar 0000H. Existe también memoria normal de lectura/escritura en el lugar 0000H, pero ésta se encuentra desconectada electrónicamente cuando la señal de RESET se activa. La CPU, desconectado el lugar 0000H, empieza a ejecutar las instrucciones de la ROM. La primera operación que realiza el programa ROM es copiarse a sí mismo en la memoria de lectura/escritura en algún lugar conveniente situado más adelante en la memoria y transferir el control de la máquina a esta copia. En este momento puede conectarse la memoria real del lugar 0000H, desconectarse la ROM y leerse el primer sector del disco.

Con cualquiera de estas técnicas, el resultado es el mismo. El primer sector del disco se lee en la memoria y el control se transfiere a la primera instrucción contenida en el sector.

El primer sector contiene el programa principal del cargador del CP/M. Este programa inicializa algunos aspectos del hardware y entonces lee en el resto de la pista 0 y en la mayoría de los sectores de la pista 1 (el número exacto depende de la longitud total del propio BIOS). El cargador del CP/M contendrá sólo una administración más primitiva de errores del disquete, intentando leer el disco una y otra vez si el hardware indica que tiene problemas al leer un sector.

El "cargador" carga el CP/M en el sitio correcto de la memoria; la dirección de carga es una constante del programa. Si se necesita construir una versión de CP/M que utilice más memoria, será necesario cambiar esta dirección de carga dentro del programa cargador, así como la dirección de salto para cuando se ha leído todo el CP/M. Esta dirección es también una constante del programa de carga.

El cargador transfiere el control a la primera instrucción del BIOS, el punto de entrada en frío. "Frío" significa que la operación empieza "en frío" a partir de una computadora vacía.

El código de carga en frío del BIOS pone a punto el hardware de la computadora. Es decir, programa los distintos chips que controlan la velocidad a la cual los puertos serie transmiten y reciben datos, inicializa los propios chips puerto-serie y, en general, prepara la computadora. Su acción final es transferir el control a la primera instrucción del BDOS con el fin de poner en marcha adecuadamente el CP/M.

Una vez que el BDOS recibe el control, se inicializa a sí mismo, examina el directorio de ficheros del disquete del sistema y transfiere el control al CCP. Entonces el CCP hace salir inmediatamente "A>" en la consola y espera la entrada de una orden; CP/M está preparado entonces para realizar las órdenes que se le den.

Al llegar a este punto, es recomendable repasar las partes de CP/M que están en memoria, comprobando cuáles son y qué funciones realizan.

Esta comprobación se dirigirá en primer lugar a la memoria. La figura 2-3 muestra las posiciones en la memoria del procesador de órdenes de consola, del sistema operativo básico de disco y del sistema básico de entrada/salida.

Al tratar estos componentes principales de la memoria —el CCP, el BDOS y el BIOS— consideraremos los módulos que interaccionan con ellos, cómo se pasan los requerimientos de acción y las funciones que pueden realizar.

Procesador de órdenes de consola

Como puede verse en la figura 2-3, el CCP es la primera parte del CP/M que "sube" a través de las direcciones de memoria, lo que es significativo cuando se considera que el CCP sólo es necesario entre programas. Cuando CP/M está a la espera (idle), necesita el CCP para interaccionar con el operador, para aceptar la próxima orden. Una vez que el CP/M ha comenzado a ejecutar la orden, el CCP está de más, ya que cualquier interacción con la consola se realizará más bien a través del programa que se está llevando a cabo que por medio del CCP. El CCP lleva, por tanto, una existencia muy desigual en la memoria. Se carga cuando se pone en marcha por primera vez el CP/M. Cuando, por medio del CCP, se solicita al

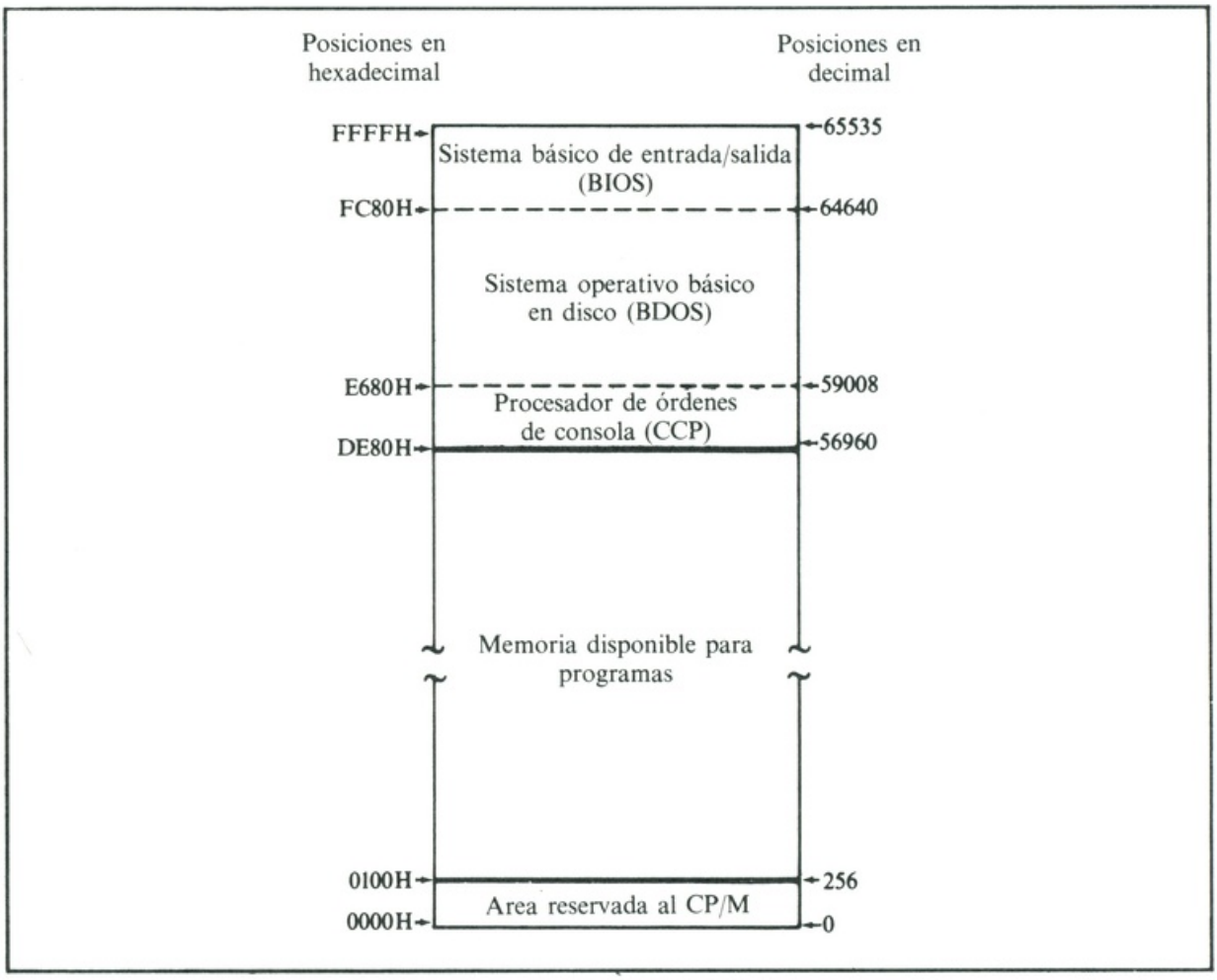


Figura 2-3. Representación de la memoria con el CP/M cargado.

CP/M que ejecute un programa, este programa puede escribirse sobre el CCP y utilizar la memoria ocupada por éste para sus propios fines. Cuando acaba el programa que se ha solicitado, el CP/M necesita volver a cargar el CCP, dispuesto para su interacción con el operador. Este proceso de recarga del CCP se conoce con el nombre de arranque en caliente. El arranque en caliente no es totalmente un "arranque en frío", sino exactamente una recarga del CCP y no se tocan ni el BDOS ni el BIOS.

¿Cómo comunica un programa al CP/M que ha terminado y que debe realizar un arranque en caliente? Saltando a la posición 0000H. En el momento en que el BIOS se inicializa a sí mismo durante la rutina de arranque en frío, coloca en la posición 0000H una instrucción de salto a la rutina de arranque en caliente, que está también en el BIOS. Una vez que la rutina de arranque del BIOS ha vuelto a cargar el CCP desde el disco,

transferirá el control al CCP. (Las rutinas de arranque en frío y en caliente se presentan más detalladas en el capítulo 6.)

Esta breve descripción indica que todas las órdenes que se dan provocan que se cargue un programa, que se escriba quizá sobre el CCP, que se ejecute el programa en cuestión y que el CCP se recargue cuando el programa salte al punto 0000H al completar su tarea. Esto no es completamente cierto, algunas de las instrucciones que se necesitan frecuentemente se encuentran en el CCP. Utilizar una de estas órdenes significa que el CP/M no tiene que cargar nada del disquete; los programas están ya en la memoria como parte del CCP. Estas órdenes, conocidas como "intrínsecas" o "residentes" o permanentes, se indican a continuación con una breve descripción de las funciones que llevan a cabo. (Todas ellas están descritas de forma más detallada en el capítulo 4.) Las órdenes "residentes" son:

DIR	Muestra los ficheros que existen en el disquete.
ERA	Borra los ficheros de un disquete.
REN	Cambia nombres de los ficheros del disquete.
TYPE	Muestra los contenidos de ficheros de texto en la consola.
SAVE	Salva parte de la memoria como fichero en el disquete.
USER	Cambia el identificador de usuario.

Sistema operativo básico de disco

El BDOS es el corazón del CP/M. El CCP y todos los programas que se utilizan con el CP/M dialogan con el BDOS para todos los contactos con el exterior. El BDOS realiza tareas tales como entrada/salida de la consola, salida de impresora, organización de ficheros (creando, borrando y volviendo a escribir ficheros y leyendo y escribiendo sectores).

El BDOS realiza todas estas cosas de una forma bastante independiente. Está relacionado únicamente con las tareas lógicas que se manejan en lugar de con la acción detallada de llevar un sector de un disquete a la memoria, por ejemplo. Estas operaciones de "bajo nivel" se realizan por el BDOS en colaboración con el BIOS.

Pero ¿cómo trabaja un programa con el BDOS? Por medio de otra instrucción de salto colocada estratégicamente en la memoria. Recuérdese que el arranque en frío colocaba el salto a la rutina de arranque caliente del BIOS en el lugar 0000H. En la posición 0005H coloca una instrucción de salto que transfiere el control a la primera instrucción del BDOS. Entonces cualquier programa que transfiera el control a la posición 0005H encontrará su camino en el BDOS. Típicamente, los programas realizan una instrucción CALL al lugar 0005H, de forma que una vez que el BDOS ha

realizado la tarea encomendada pueda volverse al programa de llamada en el sitio correcto. El programa que recurre a la ayuda del BDOS coloca valores especiales en algunos de los registros de la CPU antes de realizar la llamada a la posición 0005H. Estos valores comunican al BDOS cuál es la operación que se requiere, así como los restantes valores requeridos para la operación específica.

Sistema básico de entrada/salida

Como se ha indicado anteriormente, el BDOS trata la entrada y salida de información de forma independiente, sin preocuparse de los detalles materiales del hardware de la computadora. El BIOS es el que comunica directamente con el hardware, los puertos y los dispositivos periféricos conectados con él.

Esta separación de la entrada/salida *lógica* en el BDOS de la entrada/salida *material* del BIOS es una de las razones principales por las que el CP/M es tan popular. Significa que la misma versión del CP/M puede adaptarse a todos los tipos de computadoras, cualesquiera que sean las particularidades del diseño del hardware. Digital Research les informará de que hay más de 200.000 computadoras en el mundo que funcionan con CP/M. Casi en todas ellas funcionan copias *idénticas* del CCP y del BDOS. Unicamente el BIOS es distinto. Si se escribe un programa que funcione bajo las reglas establecidas y que interaccione únicamente con el BDOS para obtener cosas determinadas, funcionará en casi todas estas 200.000 computadoras sin que haya que cambiar una sola línea del código.

La palabra "casi" del párrafo anterior quizá ha llamado la atención. A veces los programadores pasan órdenes directamente al BIOS y no al BDOS, lo que conduce a problemas. El BIOS debe quedar fuera de los límites del programa. Es necesario saber qué es y cómo funciona, a fin de construir una versión adecuada del CP/M, pero *nunca* se deben escribir programas que dialoguen directamente con el BIOS si se desea que funcionen en otras versiones de CP/M.

Una vez planteados los peligros del diálogo con el BIOS, se puede describir la forma en que el BDOS se comunica con él. Contrariamente al BDOS, que posee un solo punto de entrada y utiliza el valor de un registro para especificar la función que se ha de realizar, el BIOS presenta varios puntos de entrada. Las primeras instrucciones del BIOS son todas puntos de entrada independientes, cada una de las cuales toma tres bytes de memoria. El BDOS introducirá al BIOS en la instrucción apropiada, según la función que se haya de realizar. Este grupo de puntos de entrada es similar en su funcionamiento a una estación de ferrocarril de mercancías. Dirige el BDOS a su destino correcto en el BIOS para la función que necesita realizar. El grupo de puntos de entrada consiste en una serie de

instrucciones JUMP, cada una de una longitud de tres bytes. El grupo, en su conjunto, se denomina tabla de saltos o vector de saltos del BIOS. El significado de cada punto de entrada se ha definido previamente. Estos puntos se detallan y comentan en el capítulo 6.

Interacciones del CCP, BDOS y BIOS

La figura 2-4 resume las funciones que realizan el CCP, el BDOS y el BIOS, la forma en que estas partes del CP/M se comunican entre sí y la forma en que un programa que funciona con el CP/M interacciona con el BDOS.

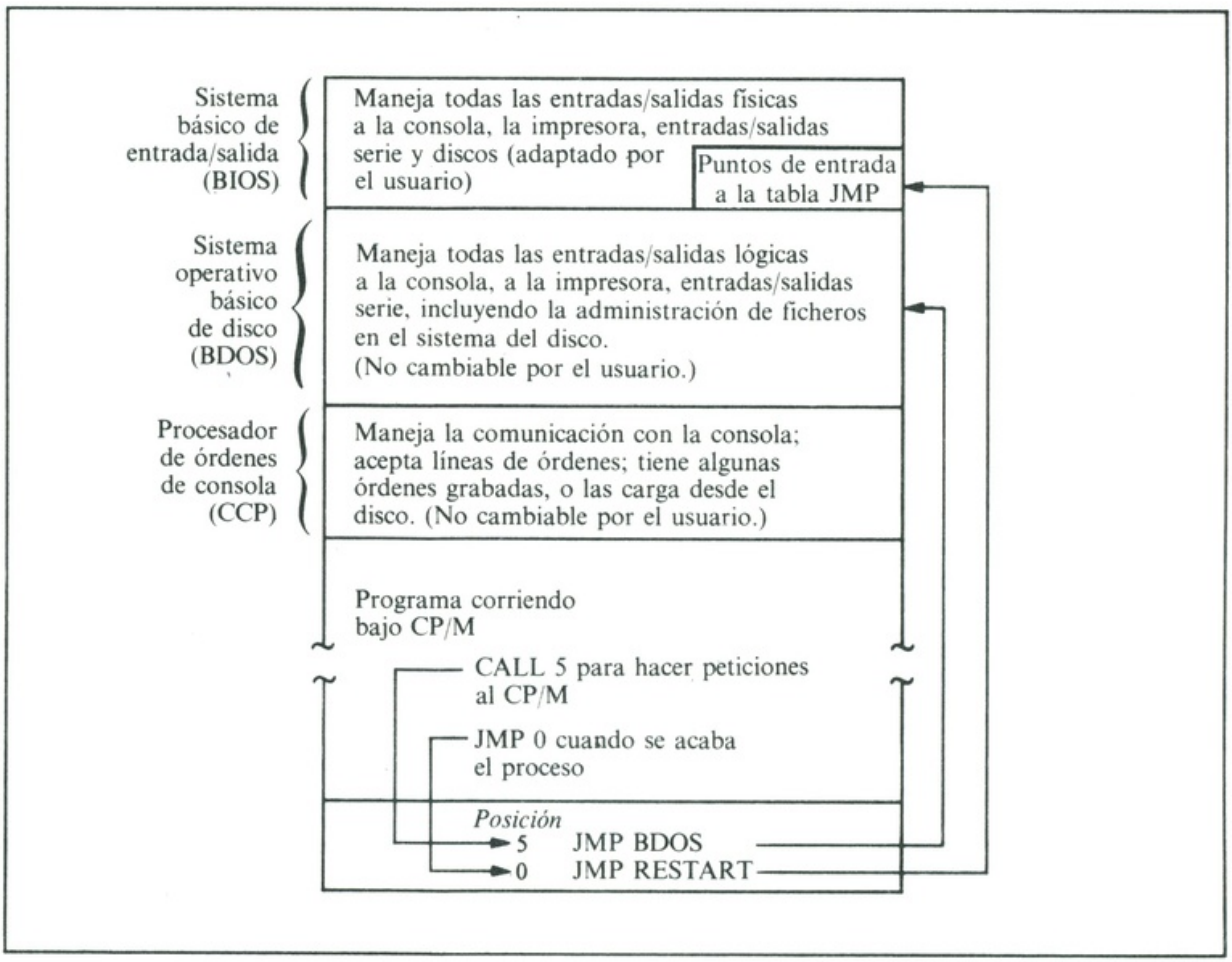


Figura 2-4. Desglose funcional de CP/M.

Cómo CP/M ve el disco
Creación de un fichero
Tablas de definición del disco
Organizaciones de ficheros

3

El sistema de ficheros del CP/M

Este capítulo proporciona una visión clara del sistema de ficheros del CP/M. El sistema operativo básico de disco (BDOS) es responsable del sistema de ficheros: Mantiene un directorio de los ficheros en el disco, anotando dónde están almacenados los datos verdaderamente en el disco. Como quiera que el sistema de ficheros retiene automáticamente esta información, pueden ignorarse los detalles de cuáles son las pistas y los sectores del disco que poseen datos de un fichero determinado.

Cómo CP/M ve el disco

Para manipular ficheros en el disco, CP/M trabaja con el disco en términos lógicos y no en términos materiales de pistas y sectores. CP/M trata al disco como si estuviera dividido en tres zonas principales.

Estas zonas se denominan: *área reservada,* que contiene el programa de

control de arranque y el propio CP/M; *directorio del fichero,* que contiene una o más entradas para cada fichero almacenado en el disco; y el *área de almacenamiento de datos,* que ocupa el resto del disco. CP/M asigna memoria a los ficheros al mismo tiempo que son creados por el programa.

El sistema básico de entrada/salida (BIOS) tiene tablas internas que indican al CP/M los tamaños respectivos de las tres áreas. Estas tablas se denominan *tablas de definición del disco,* y se describen más adelante en este mismo capítulo.

Bloques de asignación

Más bien que trabajar con los sectores individuales de 128 bytes, el CP/M reúne lógicamente algunos de estos sectores para formar un bloque. Típicamente, un bloque contendrá ocho sectores de 128 bytes (lo que hace una longitud total de 1.024 ó 1K bytes). Esto hace que la manipulación del disco sea más sencilla, ya que la magnitud de los números afectados se reduce. Por ejemplo, un disco flexible normalizado de 8 pulgadas, densidad simple y una sola cara posee 1.950 sectores de 128 bytes; los discos rígidos pueden tener 120.000 o más. Utilizando bloques que tratan ocho sectores del disco a la vez, el número de unidades de almacenamiento a utilizar se reduce sensiblemente. El número total es importante porque la información numérica se maneja como números enteros de 16 bits en los microprocesadores 8080 y Z80 y, por consiguiente, el número sin signo mayor posible es 0FFFFH (65.535 ó 64K decimal).

Cuando el CP/M se refiere a un bloque específico, todo lo que necesita es un simple número. El primer bloque es el número 0, el siguiente el número 1, etc., hasta la capacidad total restante del disco.

El bloque típico contiene 1.024 (1K) bytes, u ocho sectores de 128 bytes. Para los discos rígidos más grandes, el bloque puede ser de 16.384 (16K) bytes, que equivale a 128 sectores de 128 bytes. Al CP/M se le da la asignación por medio de una entrada en las tablas de definición del disco en el BIOS.

El tamaño del bloque no es arbitrario, sino un compromiso. El creador del BIOS que corre en el sistema —bien el fabricante o el diseñador de sistema operativo— elige el tamaño considerando la capacidad total de almacenamiento del disco. Esta elección está afectada por el hecho de que si un fichero está creado con sólo un único byte de datos en él, ése ocupará un bloque completo. Los bloques grandes pueden desperdiciar memoria del disco si hay muchos ficheros pequeños, pero pueden ser útiles cuando se exigen algunos ficheros muy grandes.

Consideremos el caso de un bloque de 1K byte. Si se crea un fichero muy pequeño que contenga nada más que un único byte de datos, se habrá ocupado un bloque completo, los 1.023 bytes restantes no se utilizarán. Se

pueden utilizar mediante la adición de nuevos datos al fichero, pero en su creación este fichero de un byte fijará un espacio muerto demasiado grande. Este es el problema: Cada fichero del disco tendrá normalmente un bloque parcialmente lleno. Si estos bloques son muy grandes, la cantidad de espacio desperdiciado (no utilizado) puede ser enorme. Un disco de 10 megabytes, con bloques de 16K bytes, puede llenarse lógicamente con todos los bloques "ocupados" con sólo 3 megabytes de datos en él.

Por otra parte, cuando se usan bloques grandes, el rendimiento del CP/M se mejora sensiblemente, ya que el BDOS se refiere al directorio con menos frecuencia. Por ejemplo, puede leer un fichero de 16K bytes con sólo una única referencia al directorio.

Por tanto, cuando se considere la asignación de bloques, deben tenerse en cuenta las siguientes cuestiones:

¿De qué tamaño es el disco lógico?
Con un disco mayor se pueden tolerar los espacios desperdiciados debidos a bloques incompletos.

¿Cuál es el tamaño medio de un fichero?
Si está previsto utilizar muchos ficheros pequeños, deben usarse bloques pequeños, de forma que se posea un "suministro" mayor de bloques. Si está previsto utilizar un número menor de ficheros de mayor tamaño, deberán usarse bloques mayores, con el fin de obtener operaciones más rápidas con el fichero.

Cuando se crea un fichero por primera vez, se le asigna un bloque único en el disco. El bloque que se le asigne depende de los otros ficheros que se tengan ya en el disco y de los bloques que ya hayan sido situados en él. CP/M mantiene una tabla de los bloques que están ocupados y de los que están disponibles. Si el fichero acumula más datos, se llenará el primer bloque. Cuando esto sucede, CP/M extenderá el fichero y situará otro bloque en él. Entonces, según crece el fichero, ocupa más bloques. No es preciso que estos bloques sean adyacentes dentro del disco y el fichero puede existir como una serie de bloques esparcidos por todo el disco. Sin embargo, si se necesita ver el fichero completo, el CP/M presenta los bloques en el orden correcto. Por tanto, los programas de aplicación pueden ignorar los bloques. CP/M conserva la información de los bloques que pertenecen a cada fichero a través del directorio de ficheros.

El directorio de ficheros

El *directorio de ficheros* está inserto entre el área reservada y el área de almacenamiento del disco. El tamaño real del directorio está definido en las tablas de definición del disco del BIOS. El directorio puede tener un número

de entradas múltiplo de dos, con una o más entradas para cada fichero que existe en el disco. Para un disquete flexible estándar de 8 pulgadas habrá sitio para 64 entradas al directorio; para un disco rígido no son raras 1.024 entradas. Cada entrada del directorio tiene una longitud de 32 bytes.

Mediante la utilización de simples operaciones aritméticas puede calcularse el espacio que ocupa el directorio en un disquete flexible. Por ejemplo, para un disco flexible la fórmula es $64 \times 32 = 2.048$ bytes $= 2$ bloques de 1.024 bytes cada uno.

Cada entrada al directorio contiene el nombre del fichero junto a una lista de los bloques utilizados actualmente por él. Naturalmente, una sola entrada al directorio de 32 bytes no puede contener todos los bloques necesarios para un fichero de 5 megabytes, especialmente debido a que el CP/M usa sólo 16 bytes del total de 32 bytes para almacenamiento de los números de los bloques.

Extensiones

A menudo hace falta que el CP/M controle ficheros que necesitan muchos bloques. Lo lleva a cabo mediante la creación de más de una entrada del directorio. En este caso las entradas segunda y subsiguientes tienen el mismo nombre de fichero que las primeras y uno de los otros bytes de la entrada del directorio se utiliza para indicar el número de secuencia de la entrada al directorio. Cada nueva entrada lleva consigo una nueva aportación de bytes que pueden ser utilizados para contener más números de bloque. En la jerga del CP/M, cada entrada del directorio se denomina extensión (extent). Comoquiera que la entrada del directorio para cada extensión tiene 16 bytes para almacenar números de bloque, puede almacenar bien 16 números de un byte o bien 8 números de dos bytes. Por consiguiente, el número total de bloques posibles en cada extensión es de 8 (para discos con más de 255 bloques) o 16 (para discos más pequeños).

Bloque de control de ficheros

Antes de que CP/M pueda hacer algo con un fichero, tiene que tener alguna información de control en la memoria. Esta información está almacenada en un *bloque de control de fichero*, o FCB. El FCB se ha descrito como un "motel" para entradas del directorio —un lugar para que residan cuando no están "en casa" dentro del disco—. Cuando las operaciones de un fichero se completan, el CP/M transforma otra vez el FCB en una entrada de directorio y vuelve a escribirlo en la entrada original. Al final de este capítulo se presenta detalladamente el FCB.

Como resumen, la figura 3-1 muestra las relaciones entre los sectores del

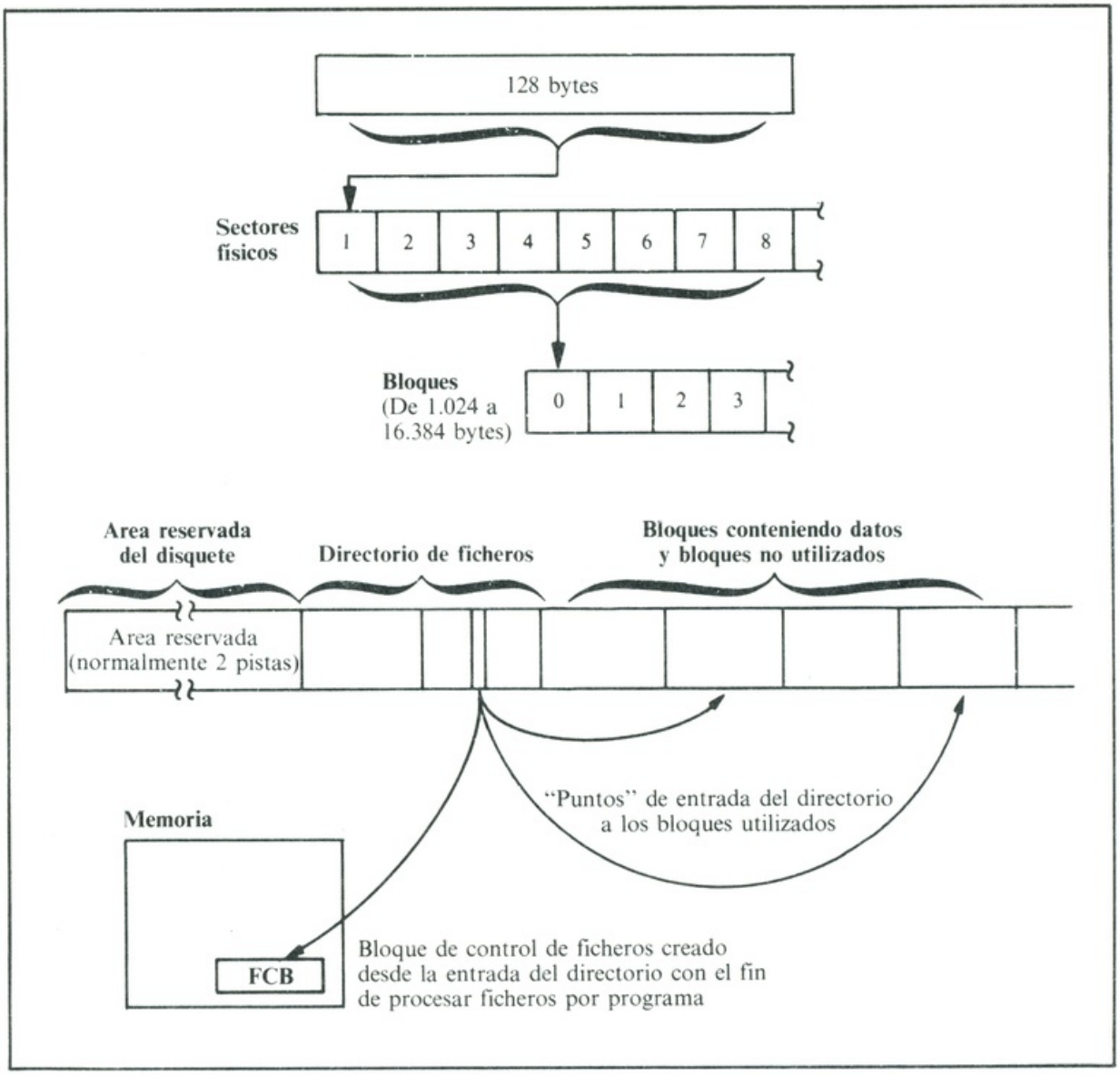


Figura 3-1. Relación jerárquica entre sectores, bloques, entradas del directorio y FCBs.

disco, los bloques, las entradas del directorio y los bloques de control de fichero.

Creación de un fichero

Para reforzar todo lo que se ha aprendido acerca del sistema de ficheros CP/M, esta sección echa un vistazo a lo que ocurre cuando un programa

que funciona con CP/M crea un fichero, escribe datos en él y lo *cierra* a continuación.

Supongamos que se ha cargado un programa en la memoria y que la CPU está a punto de ponerse a ejecutarlo. En primer lugar, el programa creará un espacio en la memoria para el FCB y colocará en él algunos valores determinados previamente, el más importante de los cuales es el nombre del fichero. El área del FCB que contenga los números del bloque según son asignados, está llena inicialmente con 0 (ceros/binarios). Como el primer bloque que está disponible para datos del fichero es el bloque 1, un número de bloque 0 significará que no se ha ocupado ningún bloque.

El programa empieza a ejecutarse. Realiza una llamada al BDOS (vía posición 0005H) solicitando que el CP/M cree un fichero. Transfiere al BDOS la dirección en memoria del FCB. Entonces el BDOS sitúa una entrada disponible en el directorio, crea una entrada nueva basada en el FCB en el programa, y vuelve al programa, preparado para escribir datos en el fichero. Obsérvese que el CP/M no se ocupa de ver si hay ya un fichero con el mismo nombre en el disco. Por esta razón, en programas más reales, una petición para abrir un fichero va precedida por una petición de borrar todos los ficheros existentes con el mismo nombre.

El programa comienza ahora escribiendo datos en el fichero, sector de 128 bytes por sector de 128 bytes. El CP/M no está preparado para escribir un byte de cada vez. Maneja los datos únicamente sector por sector, vertiendo los sectores al disco según se van llenando.

La primera vez que un programa solicita al CP/M (vía una petición BDOS) que escriba un sector en un fichero del disco, el BDOS encuentra un bloque sin utilizar y le asigna al fichero. El número del bloque se coloca dentro del FCB en memoria. Cuando todos los bloques están llenos, se encuentra uno nuevo y se le asigna, y su número se añade a la lista de bloques dentro del FCB. Finalmente, cuando el FCB no tiene más sitio para números de bloque, el BDOS:

- Escribe la entrada actualizada del directorio en el disco.
- Busca una nueva entrada disponible en el directorio.
- Vuelve a colocar el FCB en la memoria para indicar que está trabajando en ese momento en la segunda extensión del fichero.
- Borra el área de bloques del FCB y espera el sector siguiente del programa.

Entonces continúa el proceso. Se abren automáticamente nuevas extensiones hasta que el programa determina que es hora de terminar, escribe el último sector en el disco y solicita del BDOS que cierre el fichero. Entonces el BDOS convierte el FCB en una entrada final del directorio y la escribe en él.

Entrada del directorio

El directorio consiste en una serie de entradas de 32 bytes con una o más entradas para cada fichero del disco. El número total de entradas depende del formato del disco (será de 64 para un disco flexible estándar y quizá de 2.048 para un disco rígido).

La figura 3-2 muestra la estructura detallada de una entrada del directorio. Obsérvese que la descripción es realmente en código fuente del Intel 8080 para las definiciones de los datos que se necesitan para manipular una entrada del directorio. Muestra una serie de instrucciones EQU —instrucciones equivalencia usadas para asignar valores o expresiones a una etiqueta, y en este caso usadas para acceder a una entrada—. Muestra también una serie de DS o instrucciones de *definición de memoria*, usadas para declarar la memoria para una entrada. Los comentarios de cada línea describen la función de cada uno de los campos. Cuando los elementos de datos son menores de un byte, el comentario identifica los bits que se usan.

Al mirar la figura 3-2 se observa que hay en ella cierta terminología que aún no se ha usado. Esta se describe en las secciones siguientes.

Número del usuario del fichero (byte 0) Los 4 bits menos significativos (bajo orden) del byte 0 de entrada del directorio contienen un número del 0 al 15. Este es el *número de usuario* al que pertenece el fichero. Un nombre más adecuado para éste habría sido el de número de grupo de ficheros. Trabaja de la siguiente forma: Supongamos que varios usuarios comparten una computadora con un disco rígido que no puede ser extraído del sistema sin graves problemas. ¿Cómo puede estar seguro cada usuario de no enredarse

```
0000 =          FDE$USER        EQU     0       ;Número de usuario del fichero (LS 4 bits)
0001 =          FDE$NAME        EQU     1       ;Nombre de fichero (8 bytes)
0009 =          FDE$TYP         EQU     9       ;Tipo de fichero
                                                ;Sobrante de bits utilizados en el tipo
0009 =          FDE$RO          EQU     9       ;Bit 7 = 1 - Sólo lectura
000A =          FDE$SYS         EQU     10      ;Bit 7 = 1 - Modo sistema
000B =          FDE$CHANGE      EQU     11      ;Bit 7 = 0 = Fichero escrito
                                                ;
000C =          FDE$EXTENT      EQU     12      ;Número de extensión
                                                ;13, 14 reservados para CP/M
000F =          FDE$RECUSED     EQU     15      ;Registros utilizados en esta extensión
0010 =          FDE$ABUSED      EQU     16      ;Bloques utilizados
                ;
                ;
                ;
0000            FD$USER:        DS              ;Número de usuario
0001            FD$NAME:        DS      8       ;Nombre de fichero
0009            FD$TYP:         DS      3       ;Tipo de fichero
000C            FD$EXTENT:      DS      1       ;Extensión
000D            FD$RESV:        DS      2       ;Reservado para CP/M
000F            FD$RECUSED:     DS      1       ;Registros utilizados en esta extensión
0010            FD$ABUSED:      DS      16      ;Bloques utilizados
```

Figura 3-2. Declaraciones de datos para entradas de directorio de ficheros de CP/M.

con los ficheros de otros usuarios? Una solución sería que utilizase cada uno iniciales individuales como caracteres en los nombres del fichero. Así cada uno podría decir a simple vista si un fichero pertenece a otro y evitar el tocar los ajenos. Un inconveniente de este esquema es que las posiciones valorables de caracteres se usarían en el nombre del fichero, sin mencionar los inconvenientes que se generarían si varios usuarios tuviesen las mismas iniciales.

Los números de usuario del fichero están como prefijo en cada fichero y pueden considerarse en sí mismos como una parte del nombre. Cuando el CP/M se utiliza por primera vez, el usuario 0 es el usuario por defecto —el que será escogido a menos que se haya designado otro—. Cualquier fichero creado entrará en el directorio llevando el número de usuario 0. Estos ficheros se identifican como pertenecientes al área de usuario 0. Sin embargo, con un sistema de cálculo compartido hay que hacer varias modificaciones para adaptarlos a las áreas de múltiples usuarios. La orden USER hace que esto sea posible. Los números y áreas de usuario van del 0 al 15. Por ejemplo, un usuario del área 7 no podrá conseguir un directorio, acceder o borrar ficheros en el área de usuarios 5.

Este byte de número de usuario sirve también para otro propósito. Si este byte se fija al valor 0E5H, el CP/M considera que la entrada al directorio se ha borrado e ignora por completo los 31 bytes de datos restantes. El número 0E5H no se ha elegido caprichosamente. Cuando IBM definió por primera vez el estándar para los disquetes flexibles, eligió la secuencia binaria 11100101 (0E5H) como adecuada para comprobación. Un disquete nuevo formateado para ser utilizado no posee más que bytes 0E5H. Por tanto, el proceso de borrar un fichero es un borrado "lógico", en el que únicamente el primer byte de la entrada del directorio se cambia a 0E5H. Si accidentalmente se borra un fichero (y si no se ha llevado a cabo ninguna otra actividad en el directorio), puede resucitarse simplemente cambiando este primer byte a un número de usuario razonable. Este proceso se explicará en el capítulo 11.

Nombre y tipo de fichero (bytes 1-8 y 9-11) Como puede verse en la figura 3-2, el nombre de fichero en una entrada al directorio tiene una longitud de ocho bytes; y el tipo de fichero tres bytes. Estos dos campos se utilizan para denominar un fichero de forma no ambigua. El nombre de un fichero puede ser menor de ocho caracteres y el tipo del mismo menor de tres, pero en estos casos las posiciones correspondientes a caracteres no utilizados se rellenan con espacios.

Siempre que se escriban juntos el nombre y tipo de fichero, se separan con un punto. Esto no es necesario si no se utiliza el tipo de fichero (lo que equivale a decir que el tipo de fichero estará formado en su totalidad por espacios). Algunos ejemplos de nombres de ficheros son:

```
READ.ME
LONGNAME.TYP
1
1.2
```

El nombre y el tipo de fichero pueden contener caracteres de A a Z, de 0 a 9 y algunos de los llamados caracteres especiales, tales como "/" y "—". También pueden utilizarse letras minúsculas, pero hay que tener cuidado cuando se introducen órdenes en el sistema utilizando el CCP; éste convierte todas las minúsculas en mayúsculas, de forma que resulta imposible, en la práctica, encontrar ficheros que tengan letras minúsculas en la entrada del directorio. Hay que evitar la utilización excesiva de caracteres especiales. Algunos de los que se pueden usar son

```
! @ # $ % ( ) - + /
```

Los caracteres que no se deben utilizar son

```
< > . , ; : = ? * [ ]
```

Estos caracteres se utilizan por el CP/M en líneas de órdenes normales, lo que ocasionaría problemas al utilizarlos en los nombres de ficheros.

Se pueden usar caracteres raros en los nombres de ficheros si se desea. Por ejemplo, si se crean ficheros con caracteres no gráficos en sus nombres o tipos, el único modo de acceder a ellos es desde dentro de programas. No se pueden manipular estos ficheros desde el teclado, excepto utilizando nombres ambiguos (descritos en la próxima sección). Esto hace más difícil el borrado accidental, ya que no pueden especificarse los nombres directamente desde la consola.

Nombres de fichero ambiguos El CP/M está capacitado para referirse a uno o más nombres de ficheros utilizando caracteres especiales "comodín" en los nombres de fichero. El signo "?" es el principal carácter comodín. Siempre que se solicite que el CP/M realice algo relacionado con ficheros, emparejará un signo "?" con cada carácter que encuentre en el nombre del fichero. En el caso más extremo, el nombre y tipo de fichero "????????.???" se refieren a todos y cada uno de los nombres de fichero.

Para poner otro ejemplo: todos los capítulos de este libro se colocaron en ficheros denominados "CHAP1.DOC", "CHAP2.DOC", etc. Sin embargo, frecuentemente se les atribuyó la denominación "CHAP??.DOC". ¿Por qué dos signos interrogativos? Si se hubiera usado solamente uno, por ejemplo, "CHAP?.DOC", el CP/M no habría podido emparejarlo con el "CHAP10.DOC", ni con cualquier otro capítulo cuya numeración constara de dos dígitos. El emparejamiento que realiza el CP/M es estrictamente carácter por carácter.

Comoquiera que el mecanografiar signos interrogativos puede resultar aburrido y debe ponerse especial atención para saber el número exacto que se ha introducido, se dispone de una abreviación adecuada. El asterisco "*" puede utilizarse con el significado de "tantos ? como se necesiten para completar el campo del nombre o del tipo". Entonces, "????????.???" puede escribirse "*.*" y "CHAP??.DOC" puede también volverse a escribir "CHAP*.DOC".

El uso de "*" está permitido únicamente cuando se introducen nombres de fichero desde la consola. La notación del signo interrogativo, sin embargo, puede utilizarse para algunas operaciones del BDOS, con el campo nombre de fichero y tipo en el FCB puestos a "?" cuando sea necesario.

Convenios sobre el tipo de fichero Aunque se tiene plena libertad para idear nombres de fichero, los tipos de éste están sujetos a convenios y, en uno o dos casos, a las instrucciones del propio CP/M.

Los tipos que causarán problemas si no se usan correctamente son:

.ASM
Fuente en lenguaje ensamblador para el programa ASM.

.MAC
Lenguaje macroensamblador.

.HEX
Fichero hexadecimal de salida de los ensambladores.

.REL
Fichero reubicable de salida de los ensambladores.

.COM
Fichero de órdenes que es ejecutado al introducir únicamente su nombre.

.PRN
Fichero de impresión escrito en el disco cuando conviene.

.LIB
Fichero de biblioteca de programas.

.SUB
Entrada para el programa de utilidad SUBMIT del CP/M.

Los ejemplos de tipos convencionales de fichero son:

.C
Código fuente en C.

.PAS
Código fuente Pascal.

.COB
Código fuente COBOL.

.FTN
Código fuente FORTRAN.

.APL
Programas APL.

.TXT
Ficheros de texto.

.DOC
Ficheros de documentación.

.INT
Ficheros intermedios.

.DTA
Ficheros de datos.

.IDX
Ficheros de índices.

.$$$
Ficheros temporales.

El tipo de fichero es también útil para conservar algunas copias del mismo fichero; por ejemplo, "TEST.001", "TEST.002", etc.

Estados de un fichero Cada uno de los estados *sólo lectura, sistema* y *fichero modificado* requiere solamente un bit único en la entrada del directorio. Para evitar el uso de espacios innecesarios se han encajado en los tres bytes utilizados para el campo tipo de fichero. Como estos bytes están almacenados como caracteres en ASCII (que es un código de siete bits), el bit más significativo no se utiliza para el tipo de fichero y, por tanto, se encuentra disponible para indicar el estado.

El bit 7 del byte 9 muestra el estado de sólo lectura. Como indica su nombre, si un fichero está situado para ser de sólo lectura, el CP/M no permitirá que se escriba o que se borre ningún dato en él.

Si se declara que un fichero está en estado "sistema" (bit 7 del byte 10), no aparecerá cuando se muestre el directorio de ficheros. Tampoco podrá copiarse el fichero de un sitio a otro con utilidades normalizadas de CP/M, tales como PIP, a menos que se solicite específicamente. En la práctica normal se deben situar las herramientas software y los programas de

aplicación de forma que sean tanto sólo lectura como modo sistema. Sólo lectura, de forma que no se puedan borrar accidentalmente, y modo sistema, de forma que no desordenen la composición del directorio.

El bit fichero modificado (bit 7 del byte 11) está siempre puesto a 0 cuando se cierra un fichero en el cual se ha estado escribiendo. Esto puede ser útil junto con un programa de utilidad de copia de ficheros (backup) que coloca este bit a 1 cuando se hace una copia del mismo. Precisamente, examinando el directorio, este programa de utilidad puede determinar qué ficheros han sido modificados desde la última pasada y hacer la copia únicamente de aquellos que han sido modificados. Esto es mucho más fácil que recordar qué ficheros se han modificado desde que se hicieron las últimas copias.

Con un sistema de disco flexible hay menos necesidad de preocuparse de las copias fichero a fichero —es exactamente igual de sencillo copiar todo el disquete—. Este sistema es útil, sin embargo, con un sistema de disco rígido que posea cientos de ficheros almacenados en el disco.

Extensión de un fichero (byte 12) Cada entrada del directorio representa una extensión del fichero. El byte 12 de la entrada del directorio identifica el número de extensión. Si se tiene un fichero de menos de 16.384 bytes, se necesitará sólo una extensión —la número 0—. Si se escribe más información en el mismo, se necesitarán más extensiones. El número de extensión aumenta en 1 según se crea cada nueva extensión.

El número de extensión está almacenado en el directorio de ficheros porque las entradas del directorio están en secuencia aleatoria. El BDOS deberá realizar una búsqueda secuencial desde el principio del directorio para estar seguros de encontrar cualquier extensión del fichero. Si el directorio es largo, como podría ocurrir en un sistema de disco rígido, esta búsqueda puede durar segundos.

Bytes reservados 13 y 14 Estos bytes son utilizados por las partes propias del sistema de ficheros del CP/M. Desde el punto de vista del programador estarán puestos a 0.

Número de registro (byte 15) El byte 15 contiene la cuenta del número de registros (sectores de 128 bytes) que han sido utilizados en el último bloque parcialmente lleno referenciado en esta entrada del directorio. Comoquiera que el CP/M crea un fichero de forma secuencial, únicamente el bloque ocupado más recientemente no está completamente lleno.

Mapa del disco (bytes 16-31) Los bytes del 16 al 31 almacenan los números de bloque utilizados por cada extensión. Existen 16 bytes en esta área. Si el número total de bloques (como ha definido el programador en las tablas de disco del BIOS) es menor de 256, esta área puede mantener hasta 16

números de bloque. Si se ha descrito el disco conteniendo más de 255 bloques de situación, el CP/M utiliza esta área para almacenar ocho valores de dos bytes. En este caso los bloques pueden tomar valores mucho mayores.

Una entrada del directorio puede almacenar bien 8 o bien 16 números de bloque. Si el fichero no se ha extendido lo suficiente para requerir este número total de bloques, las posiciones no utilizadas de la entrada se rellenan con ceros. Puede pensarse que esto creará problemas, por lo que parece que varios ficheros serán situados una y otra vez en el bloque 0. En realidad no existe problema porque el propio directorio de ficheros ocupa siempre el bloque 0 (y, dependiendo de su tamaño, algunos de los bloques siguientes). Para todos los propósitos prácticos, el bloque 0 "no existe", por lo menos para el almacenamiento de datos.

Obsérvese que si accidentalmente se pierde la relación entre los ficheros y sus bloques —por ejemplo, si los datos de un bloque determinado se escriben más allá de su lugar asignado, o bien si dos o más entradas activas del directorio contienen el mismo número de bloque—, el CP/M no puede acceder a la información de manera adecuada y el disco se inutiliza.

Algunos programas de utilidad disponibles comercialmente manipulan el directorio. Se pueden usar para inspeccionar y cambiar el directorio dañado, reviviendo ficheros borrados si es necesario. Existen otros programas de utilidad que se pueden utilizar para anular sectores inservibles del disco. Estos programas encuentran las áreas inservibles, trabajan hacia atrás a partir de los números de pista y de sector, y calculan el bloque en el que ocurre el error. Una vez se conocen los números del bloque, crean un falso fichero, bien en el área de usuario 15 o, en algunos casos, en un área de usuario "imposible" (una mayor de 15), que parece "apropiarse" de todos los bloques inservibles.

Un buen programa de utilidad protege la integridad del directorio verificando que cada bloque de situación está "poseído" únicamente por una entrada de directorio.

Tablas de definición del disco

Como se ha mencionado previamente, el BIOS contiene tablas que indican al BDOS cómo ha de ver las unidades de disco que forman parte del sistema de cálculo. Estas tablas estarán construidas *por el programador*. Si se utilizan disquetes normalizados de 8 pulgadas, de una sola cara, de simple densidad, pueden usarse los ejemplos del manual de Digital Research *Guía de modificaciones de CP/M 2*. Pero si usa algún otro sistema más complicado hay que seguir criterios más cuidadosos. Cualquier error en las *tablas de definición del disco* puede crear problemas graves, especialmente

cuando se intenta corregir disquetes creados utilizando las tablas erróneas. El programador deberá asegurarse de la corrección de las tablas poniendo el máximo cuidado.

Un último detalle antes de entrar en la estructura de la tabla: El hecho de que las tablas existan y definan un "patrón" particular del disco no significa que este disco tenga que estar necesariamente conectado al sistema. Las tablas describen los discos *lógicos* y no existe forma alguna para el hardware de controlar si las tablas del disco son correctas. Puede tenerse una computadora con un solo disco duro; entonces describir el disco como si estuviera dividido en varios discos *lógicos*. El CP/M visualizará cada uno de tales "discos" independientemente y deberán ser considerados como discos separados.

Tabla cabecera de parámetros del disco

Esta tabla es el punto de partida en las tablas de definición del disco. Es la estructura más alta y no contiene más que las direcciones de otras estructuras. Existe una entrada en esta tabla para cada disco lógico que se quiera describir. Existe un punto de entrada en el BIOS que devuelve la dirección de la tabla cabecera de parámetros para un disco lógico específico.

Un ejemplo del código necesario para definir una tabla cabecera de parámetros del disco se muestra en la figura 3-3.

Desplazamiento de sectores Para definir el desplazamiento de sectores, llamado también entrelazamiento de sectores, imagínese un disquete girando en la unidad del disco. Los sectores de la pista sobre la cual se encuentra la cabeza están pasando por ésta uno tras otro —sector 1, sector 2, etc.— hasta que el disquete ha girado en su totalidad. Entonces la secuencia se repite. Un disquete estándar de 8 pulgadas posee 26 sectores en cada pista y el disco gira a 360 rpm. Un giro del disquete dura 60/360 segundos, aproximadamente 166 milésimas de segundo por pista, o 6 milésimas de segundo por sector.

Imagínese ahora al CP/M cargando un programa para tal disquete. El BDOS toma un número finito de tiempo para leer y tratar cada sector, ya que lee un solo sector a la vez. Tiene que realizar repetidamente lecturas para cargar un programa. En el momento en que el BDOS ha leído y cargado el sector n, será demasiado tarde para leer el sector n+1. Este sector ya habrá pasado por la cabeza y no volverá a pasar hasta 166 milésimas de segundo después. Procediendo de esta forma, se necesitan casi cuatro segundos y medio para leer una pista completa.

Este problema puede resolverse mediante la simple numeración de sectores *lógicamente*, de forma que haya algunos sectores materiales entre cada sector lógico. Este procedimiento, denominado desplazamiento de

```
                DPBASE:                         ;Base de la cabecera de parámetros
                                                ; (usada para acceder a las cabeceras)
0000 1000               DW      SKEWTABLE       ;Apuntador a la tabla de conversión
                                                ; de sectores lógicos a físicos
0002 0000               DW      0               ;Area de anotaciones usada por CP/M
0004 0000               DW      0
0006 0000               DW      0
0008 2A00               DW      DIRBUF          ;Apuntador al área de trabajo de
                                                ; la memoria de directorio
000A AA00               DW      DPB0            ;Apuntador al bloque de parámetros del disco
000C B900               DW      WACD            ;Apuntador al área de trabajo (utilizada
                                                ; para comprobación de cambios de disquetes)
000E C900               DW      ALVEC0          ;Apuntador al vector de situación
                ;
                ;
                ;       Las equivalencias siguientes podrían deducirse de los valores que
                ;       se encuentran en el bloque de parámetros del disco.
                ;       Se muestran aquí sólo para completar.
                ;
003F =          NODE    EQU     63              ;Número de entradas al directorio
00F2 =          NOAB    EQU     242             ;Número de bloques
                ;
                ;       Ejemplo de definiciones de datos para aquellos objetos
                ;       apuntados por los parámetros cabecera del disco
                ;
                SKEWTABLE:                      ;Tabla de desplazamiento de sectores
                                                ; indexada por sectores lógicos
0010 01070D13           DB      01,07,13,19     ;Sectores lógicos 0,1,2,3
0014 19050B11           DB      25,05,11,17     ;4,5,6,7
0018 1703090F           DB      23,03,09,15     ;8,9,10,11
001C 1502080E           DB      21,02,08,14     ;12,13,14,15
0020 141A060C           DB      20,26,06,12     ;16,17,18,19
0024 1218040A           DB      18,24,04,10     ;20,21,22,23
0028 1016               DB      16,22           ;24,25
                ;
002A            DIRBUF: DS      128             ;Memoria intermedia de directorio
00AA            DPB0:   DS      15              ;Bloque de parámetros del disco
                                                ;Esta es normalmente una tabla
                                                ; de constantes.
                                                ;Muestra de una definición
                                                ; ficticia
00B9            WACD:   DS      (NODE+1)/4      ;Area de trabajo para comprobación de directorio
                                                ;Se usa únicamente para discos transportables
00C9            ALVEC0: DS      (NOAB/8)+1      ;El vector de situación #0
                                                ;Necesita 1 bit
                                                ; por bloque
```

Figura 3-3. Declaraciones de datos para una cabecera de parámetros de disco.

sectores o entrelazamiento de sectores, se muestra en la figura 3-4. Nótese que, a diferencia de los sectores materiales, los sectores lógicos están numerados de 0 a 25.

La figura 3-4 muestra el entrelazamiento de sectores estándar del CP/M para disquetes de 8 pulgadas, de una cara y de simple densidad. Puede apreciarse que el sector lógico 0 tiene seis sectores entre él y el sector lógico 1. Existe un hueco similar entre cada uno de los sectores lógicos, de forma que existen seis "tiempos de sector" (aproximadamente 38 milésimas de segundo) entre dos sectores lógicos adyacentes, lo que da tiempo suficiente para que el software tenga acceso a cada sector. Son necesarias varias vueltas completas del disco para leer todos los sectores por orden. En la figura 3-4 las columnas verticales de los sectores lógicos muestran cuáles son los sectores que se leen en cada una de las vueltas sucesivas.

Sector físico	Sector lógico					
	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6
1	0					
2				13		
3			9			
4						22
5		5				
6					18	
7	1					
8				14		
9			10			
10						23
11		6				
12					19	
13	2					
14				15		
15			11			
16						24
17		7				
18					20	
19	3					
20				16		
21			12			
22						25
23		8				
24					21	
25	4					
26				17		

NOTA: Sector adicional entre los sectores lógicos 12 y 13.

Figura 3-4. Desplazamiento de sectores físicos a lógicos.

Un entrelazamiento erróneo puede afectar profundamente al rendimiento. No es un efecto gradual; si se "yerra" en el entrelazamiento, se percibirá un rendimiento muy lento. En el ejemplo que se cita se necesitan seis vueltas del disquete para leer toda la pista —esto dura un segundo, en oposición a los cuatro y medio que dura sin entrelazamiento—. Pero no debe imaginarse que se puede cambiar impunemente el entrelazamiento; los ficheros escritos con un entrelazamiento permanecen en la misma forma. Hay que estar seguros de que se leen con el mismo entrelazamiento con el que fueron escritos.

Algunos controladores de disco pueden simplificar este procedimiento. Cuando se formatea el disquete se pueden escribir las direcciones de los sectores en el disquete, con el entrelazamiento construido ya en él. Cuando el CP/M solicita el sector n, la parte electrónica del controlador espera hasta que ve pasar la cabecera del sector requerido. Entonces inicia la operación de lectura o escritura. En este caso se puede encajar el entrelazamiento correctamente en el formateado del disquete.

Comoquiera que un entrelazamiento erróneo produce un rendimiento

malísimo, es fácil saber cuándo se ha elegido el correcto. Algunos programadores utilizan el tiempo requerido para formatear un disquete como criterio de rendimiento para optimizar el entrelazamiento. Esta práctica no es correcta, ya que en circunstancias normales se empleará poquísimo tiempo en el formateado de disquetes. El tiempo que se tarda en cargar programa será un árbitro más exacto, ya que se requiere mucho más tiempo para esta operación. Podría decirse que actualizar un fichero es más representativo, pero, en realidad, muchas actualizaciones producen una actividad del disco lenta y esporádica. Esta forma de utilización del disco no es, pues, conveniente para colocar el entrelazamiento correcto.

Los discos rígidos no presentan problemas para el entrelazamiento. Giran a 3.600 rpm o más, y a esta velocidad no puede servir ningún entrelazamiento. Se pueden usar algunos trucos para mejorar el rendimiento de un disco rígido —éstos se tratarán en la sección denominada "Consideraciones especiales para discos rígidos", más adelante en este mismo capítulo.

Para entender mejor estas teorías, estudiaremos un ejemplo de la tabla de entrelazamientos normalizados, o *tabla de deslizamiento*. No debe olvidarse que al programa que accede a esta tabla se le dará, en primer lugar, un sector *lógico*. Por consiguiente, tiene que volver el sector físico adecuado.

La figura 3-5 muestra el programa para la tabla desplazamientos y el programa que puede utilizarse para acceder a ella. Esta está provista de un

```
                SKEWTABLE:                          ;Sector lógico
0000 01070D13           DB      01,07,13,19         ;0,1,2,3
0004 19050B11           DB      25,05,11,17         ;4,5,6,7
0008 1703090F           DB      23,03,09,15         ;8,9,10,11
000C 1502080E           DB      21,02,08,14         ;12,13,14,15
0010 141A060C           DB      20,26,06,12         ;16,17,18,19
0014 1218040A           DB      18,24,04,10         ;20,21,22,23
0018 1016               DB      16,22               ;24,25
                ;
                ;       El programa de traducción de sectores lógicos
                ;        a sectores físicos es el siguiente:
                ;
                ;       A la entrada, el sector lógico será transferido al
                ;       registro BC desde CP/M como un número de 16 bits
                ;       CP/M transfiere también la dirección de la tabla
                ;       de desplazamiento (la encuentra buscando en la cabecera de entrada de
                ;       parámetros del disco).
                ;
                ;       A la salida, el sector físico se sitúa
                ;       en los registros HL.
                ;
                SECTRAN:
001A EB                 XCHG            ;HL -> dirección base tabla de desplazamientos
001B 09                 DAD     B       ;HL -> entrada por sector físico
                                        ;       en la tabla de desplazamientos
001C 6E                 MOV     L,M     ;L = sector físico
001D 60                 MOV     H,0     ;HL = sector físico
001E C9                 RET             ;Retorno de control al BDOS
```

Figura 3-5. Declaraciones de datos para la tabla de enlazamiento para disquetes normalizados.

índice de sectores lógicos y la correspondiente entrada de tabla es el sector físico. Puede comprobarse que el programa considera que el primer sector *lógico* asignado por el CP/M es el sector número 0. Luego no hay necesidad de restar 1 del número del sector antes de utilizarlo como subíndice de la tabla.

Areas no utilizadas en la tabla cabecera de parámetros del disco Las tres palabras presentadas como ceros en la figura 3-3 son usadas por el CP/M como variables temporales durante las operaciones del disco.

Memoria intermedia de directorio (Directory Buffer) (DIRBUF) La *memoria intermedia de directorio* es un área de 128 bytes usada por el CP/M para almacenar un sector del directorio mientras se procesan las entradas del mismo. Se necesita únicamente una memoria intermedia, ya que puede ser compartida por todos los discos lógicos del sistema.

Bloque de parámetros del disco (DPB0) El *bloque de parámetros del disco* describe las características particulares de cada disco lógico. En general, se necesitará un bloque de parámetros separado para cada *tipo* de disco lógico. Los discos lógicos pueden compartir un bloque de parámetros únicamente si sus características son idénticas. Se puede usar, por ejemplo, un único bloque de parámetros para describir todas las unidades de disquete de una sola cara y simple densidad que se tengan en el sistema. Sin embargo, se necesitaría otro bloque de parámetros para describir las unidades de disquetes de dos caras y doble densidad. También es difícil compartir bloques de parámetros cuando un disco rígido está dividido en varios discos lógicos. El motivo se entenderá más tarde al contemplar los contenidos de un bloque de parámetros que se describe más adelante en este mismo capítulo.

Area de trabajo para comprobar cambios de disquetes (WACD) Uno de los mayores problemas con que se enfrenta el CP/M cuando trabaja con medios transportables como disquetes es que el operador de la computadora, sin ningún tipo de aviso, puede abrir la unidad de disquetes y sustituirlo por un disquete diferente. En las versiones primitivas del CP/M el resultado era que en el disquete insertado nuevamente se escribían datos sobre los que ya contenía el disquete original.

Con la versión actual del CP/M puede pedirse que el CP/M controle si el disquete ha sido cambiado. Después de esta solicitud, el CP/M examina las entradas del directorio con las que ha trabajado y, si detecta que el disquete ha sido cambiado, declara que el disquete completo está en condiciones de sólo lectura e inhibe cualquier escritura posterior en el mismo. Esta situación tendrá efecto hasta que se realice la siguiente operación de arranque caliente. Un arranque caliente se realiza cuando termina un

programa o cuando se introduce un CONTROL-C en el CCP, reinicializando el sistema operativo.

El valor del WACD es la dirección de una memoria intermedia o área de almacenamiento temporal, que el CP/M puede utilizar para comprobar el directorio. La longitud de esta memoria intermedia está definida (algunas veces fuera de lugar) en el bloque de parámetros de disco.

Vector de situación (ALVEC0) El CP/M visualiza cada disco como un conjunto de bloques, asignando bloques a los ficheros individuales según éstos van siendo creados o expandidos, así como ir abandonando bloques según se van borrando los ficheros.

El CP/M necesita algún mecanismo para mantener información de los bloques que se usan y los que están libres. Utiliza el *vector de situación* para formar un *mapa de bits,* donde cada bit del mapa corresponde a un bloque específico. El bit más significativo (el 7) del primer byte corresponde al primer bloque, número 0. El bit 6 corresponde al bloque 1, y así sucesivamente para todo el disco.

Siempre que se requiere del CP/M que utilice un disco lógico, el CP/M *examinará* el disco. Esto consiste en leer el directorio de ficheros y, para cada entrada o extensión activas, interaccionar con los bloques que "posee" la entrada o extensión de que se trate. Para cada número de bloque de la extensión, el bit correspondiente del vector de situación se pone a 1. Al final de este proceso, el vector de situación representará cuidadosamente un mapa sobre los bloques que están en uso y los que están libres.

Cuando el CP/M está buscando un bloque de situación no utilizado, trata de encontrar uno próximo al último utilizado, con el fin de evitar que los ficheros queden demasiado fragmentados.

Con el fin de reservar espacio suficiente para el vector de situación, es preciso reservar un bit por cada bloque. El cálculo del número de bloques se trata en la sección "Número máximo de bloques", que se encuentra más adelante en este mismo capítulo.

Bloque de parámetros del disco

El *bloque de parámetros del disco* en versiones anteriores del CP/M estaba construido en el BDOS y era un secreto celosamente guardado del sistema de ficheros del CP/M. Para hacer el CP/M adaptable a sistemas de disco rígido, Digital Research decidió desplazar el bloque de parámetros al BIOS, en el que cualquiera puede adaptarlos. Debido a la naturaleza particular del sistema de ficheros del CP/M, se encontrarán aun algunos espacios aparentemente sobrantes y la explicación que se da aquí puede parecer algo superficial. Sin embargo, la falta de aclaraciones no disminuye en absoluto la capacidad para utilizar el CP/M como herramienta.

La figura 3-6 muestra el programa necesario para definir un bloque de parámetros para disquetes de 8 pulgadas, de una sola cara. Esta tabla está apuntada por una entrada en la tabla cabecera de parámetros del disco —es decir, su dirección está dada en ésta—. Cada una de las entradas que se muestran en el bloque de parámetros del disco está explicada en las secciones siguientes.

Sectores por pista Es el número de sectores de 128 bytes por pista. El disquete normalizado presentado en el ejemplo tiene 26 sectores. Como puede verse, el hecho de indicar que el CP/M tiene 26 sectores por pista no aclara si el primer sector está numerado 0 ó 1. El CP/M supone que el primer sector es 0; se encomienda a una subrutina de traducción de sectores el que halle a qué sector material corresponde.

Los discos rígidos tienen normalmente un tamaño de sector mayor de 128 bytes. Esto se trata en la sección de consideraciones para discos rígidos.

Desplazamiento de bloques, enmascaramiento y máscara de extensión Estos campos de nombre misterioso se utilizan dentro de CP/M durante las operaciones de ficheros del disco. Los valores que se especifiquen para ellos dependen, en primer lugar, del tamaño de bloque que se desee.

El tamaño de bloque puede variar de 1.024 (1K) a 16.384 (16K) bytes. Existe un término medio entre estos dos extremos, como se indicó en la sección de bloques, al principio de este capítulo.

Se requiere un tamaño de bloque de 1.024 (1K) bytes para disquetes con capacidades de más de un megabyte, y un tamaño de bloque de 4.096 (4K) bytes para disquetes mayores o discos rígidos.

Si se puede definir el tamaño de bloque que se desea usar, pueden seleccionarse entonces los valores para el desplazamiento de bloques y el enmascaramiento de bloques en la tabla 3-1.

```
                DPB0:
0000 1A00               DW      26          ;Sectores por pista
0002 03                 DB      3           ;Desplazamiento de bloques
0003 07                 DB      7           ;Máscara de bloques
0004 03                 DB      3           ;Máscara de extensión
0005 F200               DW      242         ;Número máximo de bloque
0007 3F00               DW      63          ;Número de entradas al directorio-1
0009 C0                 DB      1100$0000B  ;Mapa de bits para los bloques usados
000A 00                 DB      0000$0000B  ; por el directorio
000B 1000               DW      16          ;Número de bytes en la memoria de comprobación del directorio
000D 0200               DW      2           ;Número de pistas antes del directorio
```

Figura 3-6. Declaraciones de datos para el bloque de parámetros del disco para disquetes normalizados.

Tabla 3-1. Valor del desplazamiento y el enmascaramiento de bloques.

Tamaño del bloque	Desplazamiento de bloque	Máscara de bloques
1.024	3	7
2.048	4	15
4.096	5	31
8.192	6	63
16.384	7	127

Hay que seleccionar el tamaño del bloque requerido en la columna de la izquierda, la tabla indica entonces los valores de desplazamiento y enmascaramiento de bloque que hay que introducir en el bloque de parámetros del disco.

La última de estas tres variables, la *máscara de extensión*, depende no sólo del tamaño del bloque, sino también de la capacidad total de memoria del disco lógico. En cuanto a esta última consideración sólo es importante para el cálculo el hecho de que haya o no menos de 256 bloques en el disco lógico. Para comprobarlo únicamente hay que dividir la capacidad del disco lógico por el tamaño del bloque elegido y ver si resultan menos de 256 bloques.

Manteniendo en la mente esta respuesta y el tamaño del bloque, consúltese la tabla 3-2 para obtener el valor adecuado para el campo de máscara de extensión del bloque de parámetros. Selecciónese la línea adecuada según el tamaño del bloque elegido. Entonces, según el número total de bloques del disco lógico, selecciónese la máscara de extensión en la columna correspondiente.

Número máximo de bloques Este valor es el *número* del último bloque del disco lógico. Como el primer número de bloque es 0, este valor es *menor en una unidad* que el número total de bloques del disco. Cuando exista únicamente un bloque parcial, el número de bloques se redondea por defecto.

Tabla 3-2. Valores de la máscara de extensión.

Tamaño del bloque	Número de bloques	
	1 a 255	De 256 en adelante
1.024	0	(Imposible)
2.048	1	0
4.096	3	1
8.192	7	3
16.384	15	7

La figura 3-7 presenta un ejemplo para disquetes de 8 pulgadas, de una sola cara y simple densidad. Obsérvese que el CP/M utiliza dos pistas reservadas en el formato de este disquete.

Número de entradas del directorio menos 1 No debe confundirse esta entrada con el número de ficheros que pueden almacenarse en el disco lógico; es únicamente el número de *entradas* (menos una). Cada extensión de cada fichero toma una entrada del directorio, por lo que ficheros muy grandes consumirán bastantes entradas. Obsérvese también que el valor en la tabla es *menor en una unidad* que el número de entradas.

En un disquete normalizado de 8 pulgadas, el valor es de 63 entradas. En un disco rígido, puede desearse utilizar 1.023 o bien 2.047. Recuérdese que el CP/M realiza un examen secuencial a través del directorio y que éste lleva una cantidad apreciable de tiempo. Por tanto, habrá que equilibrar el número de discos lógicos con la estimación del tamaño del mayor fichero que se desee soportar.

Como nota final, habrá que asegurarse de elegir un número de entradas que se ajuste igualmente a uno o más bloques. Cada entrada del directorio necesita 32 bytes, de forma que puede calcularse el número de bytes requerido. Hay que asegurarse de que este número sea divisible por el tamaño del bloque que se ha elegido sin que quede ningún resto.

Bloques para el directorio Este es un valor raro; no es un número, sino un mapa de bits. Al mirar la figura 3-6 se observa que el valor ejemplo está totalmente escrito en binario, para explicar cómo se define este valor. Este valor de 16 bits posee un bit puesto a 1 para cada bloque que ha de utilizarse para el directorio.

Características físicas:		**Calcular:**	
77	Pistas/disquete	77	Pistas/disquete
26	Sectores/pista	− 2	Pistas reservadas para CP/M
128	Bytes/sector	75	Pistas para almacenamiento de ficheros
2	Pistas reservadas para CP/M	×26	Número de sectores
1.024	Bytes/bloque	1.950	Sectores para almacenamiento de ficheros
		×128	Bytes por sector
		249.600	Bytes para almacenamiento de ficheros
		÷1.024	Bytes/bloque
		243,75	Número total de bloques
		242	Número del último bloque (redondeado y basado en que el primer bloque es el bloque 0)

Figura 3-7. Cálculo del número máximo de bloques para disquetes normalizados.

Este valor se calcula a partir del número de entradas del directorio que se desea tener en el disco y del tamaño del bloque que se desea usar. Una constante en este cálculo es que el tamaño de todas las entradas del directorio es de 32 bytes.

En el ejemplo se requieren 64 entradas (recuérdese que el número indicado es uno menos que el valor requerido). Cada entrada tiene 32 bytes. El número total de bytes requerido por el directorio es entonces de 64 veces 32, o sea, 2.048 bytes. Dividiendo éste por el tamaño del bloque, que es 1.024, resulta que hay que reservar dos bloques para el directorio. En el valor del ejemplo puede verse que se colocan los dos bits más significativos del valor de 16 bits.

Una advertencia: no debe caerse en la tentación de declarar este valor utilizando una pseudo operación DW (definición de palabra). Al actuar así se almacena el valor *invertido por bytes.*

Tamaño de la memoria intermedia para comprobación del directorio Como se ha indicado anteriormente, al tratar sobre la cabecera de parámetros del disco, el CP/M puede ser utilizado para comprobar las entradas del directorio siempre que esté trabajando en éste. Para hacer esto, el CP/M necesita un área de comprobación, denominada *área de trabajo para comprobar los disquetes cambiados,* o WACD, en la cual puede mantenerse trabajando variables que conservan un registro comprimido de lo que está en el directorio. La longitud de esta área de memoria intermedia se conserva en el bloque de parámetros del disco; su dirección se especifica en la cabecera de parámetros. Comoquiera que el CP/M conserva una grabación comprimida del directorio, se necesita solamente proveer un byte para cada cuatro entradas al directorio. Como puede verse en la figura 3-6, hay 16 bytes para mantener información de las 64 entradas del directorio.

Número de pistas antes del directorio La figura 3-8 muestra la disposición del CP/M en un disquete. Como puede verse, las dos primeras pistas están reservadas, conteniendo el código inicial de arranque en frío y al propio CP/M. El ejemplo de la figura 3-6, que presenta el código para un disco flexible normalizado, muestra dos pistas reservadas (número de pistas reservadas antes del directorio).

Este valor de *desplazamiento de pistas,* como se denomina a veces, procura un método conveniente para la división de un disco físico en varios discos lógicos.

Consideraciones especiales para discos rígidos

Si se desea que el CP/M trabaje con un disco rígido, hay que procurarse un programa y construir tablas que hagan que el CP/M trabaje como si

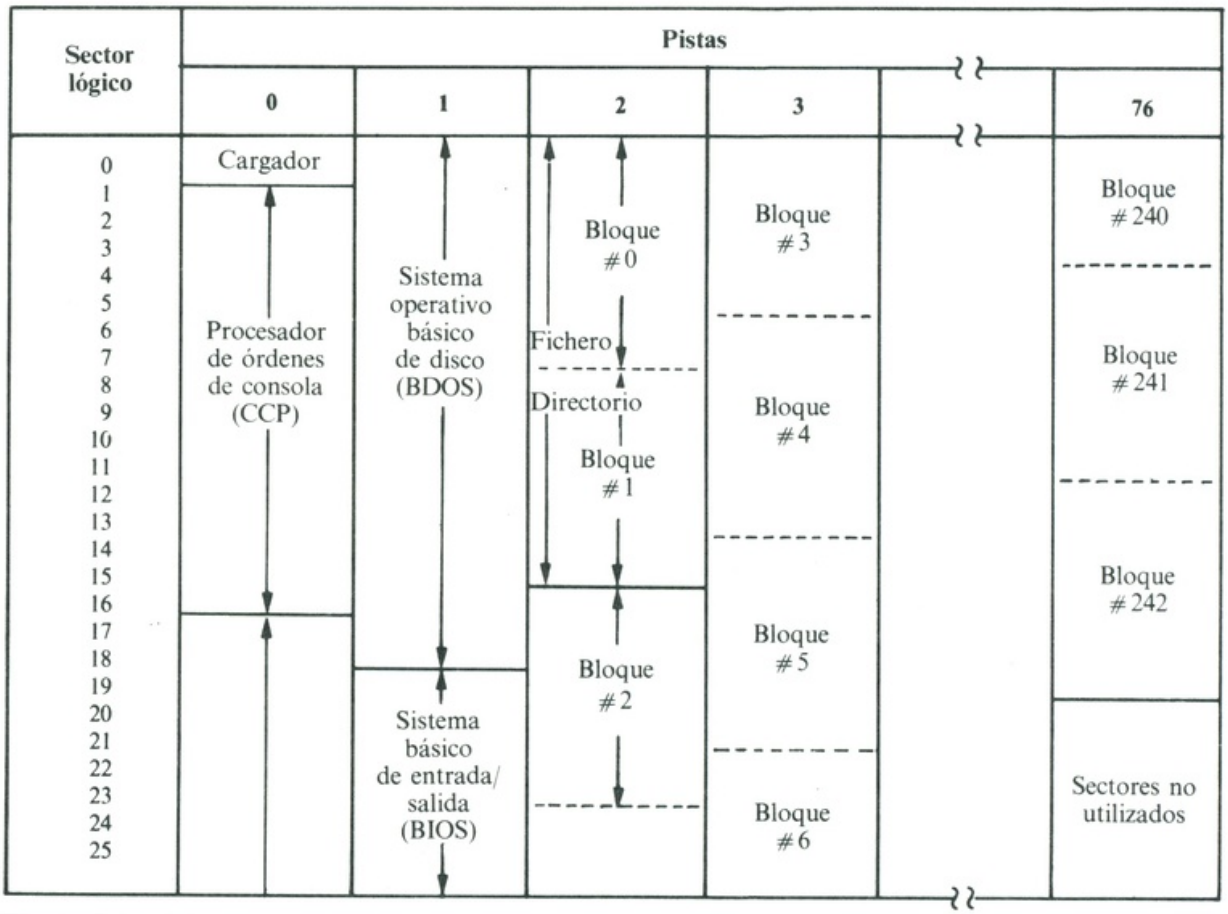


Figura 3-8. Disposición de un disquete normalizado.

estuviera funcionando con un disco flexible muy amplio. También habrá que incluir sectores de 128 bytes. Sin embargo, esto no presenta dificultad alguna.

Para adaptar discos rígidos al tamaño de sector de 128 bytes, hay que proveerse de un programa en el controlador de disco en el BIOS, de manera que parezca que se está leyendo y escribiendo en sectores de 128 bytes, aunque esté trabajando realmente en sectores de 512 bytes. Este programa se denomina rutina de *bloqueo/desbloqueo*.

Si los discos rígidos tienen tamaños de sector distintos de 128 bytes, ¿cuál habrá de ser el número de sectores por pista y cuál el número de pistas?

Los discos rígidos existen en todos los tamaños. La situación es más confusa en los controladores de disco, el hardware que controla el disco. En muchos casos se puede considerar el disco como una serie de sectores sin ninguna clase de pistas. El controlador, dando un número de sector *relativo* por medio del BIOS, puede traducir este número de sector en: la pista, la

cabeza de lectura/escritura (si existe más de una) y el sector que se esté utilizando prácticamente.

Además, muchos discos rígidos giran tan rápidamente que no se gana nada utilizando un algoritmo de desplazamiento de sectores. No existe ningún medio de leer más de un sector físico por revolución; no hay tiempo suficiente.

En muchos casos es deseable dividir un disco rígido único en varios discos lógicos más pequeños, lo que se hace principalmente por razones de rendimiento: Discos pequeños, con directorios menores, se traducen en operaciones de fichero más rápidas.

La cabecera de parámetros del disco tendrá ceros para la entrada desplazamiento de sectores y el apuntador para la memoria intermedia WACD. En general, los discos rígidos *no pueden* cambiarse, al menos no sin desconectar el aparato y cambiar toda la unidad de disco. Si se utiliza uno de los discos de la nueva generación que son discos rígidos transportables, se necesitará utilizar la comprobación de directorio de CP/M.

El bloque de parámetros del disco para un disco rígido será muy diferente del usado para un disquete. El número de sectores por pista necesita ser cuidadosamente considerado. Recuérdese que este es el número de sectores de 128 bytes. La conversión del tamaño del sector físico a sectores de 128 bytes se hará en el controlador del disco en el BIOS.

Si se tiene un controlador de disco que trabaja en términos de sectores y pistas, todo lo que se necesita es calcular el número de sectores de 128 bytes en cada pista. Multiplicando el número de sectores físicos por pista por su tamaño en bytes y dividiendo entonces el resultado por 128 se obtiene el número de sectores de 128 bytes por cada pista física.

Pero ¿qué ocurre con los controladores que visualizan sus discos rígidos como una serie de sectores sin referencia a pista alguna? Oscurecen el hecho de que los sectores están colocados en pistas concéntricas sobre la superficie del disco. En este caso puede usarse un truco en el CP/M. Puede ponerse el valor de los "sectores por pista" al número de sectores de 128 bytes que se encajan dentro de los sectores físicos del disco. Para realizar esto hay que dividir el tamaño del sector físico entre 128. Por ejemplo, un tamaño de sector físico de 512 bytes dará un resultado de cuatro sectores de 128 bytes por "pista". Ahora podrá visualizarse el disco duro como si se tuvieran tantas "pistas" como sectores físicos. Utilizando este método se evita tener que realizar toda clase de operaciones aritméticas en los números de sector de CP/M; el número de "pista" que CP/M solicitará al BIOS para mover las cabezas de disco será el *sector físico relativo*. Una vez que el controlador ha leído este sector físico, puede mirarse el número de sector de 128 bytes, que será 0, 1, 2 ó 3 (para un sector físico de 512 bytes) con el fin de seleccionar cuáles de los 128 bytes es necesario sacar o introducir en la memoria intermedia del disco.

El desplazamiento de bloques, la máscara de bloque y la máscara de

extensión se calcularán como se ha indicado anteriormente. Hay que utilizar un tamaño de bloque de 4.096 bytes. Esto dará un valor de 5 para el desplazamiento, 31 para la máscara de bloque y, teniendo en cuenta que se tendrán más de 256 bloques para cada disco lógico, un valor de 1 para la máscara de extensión.

El número máximo de bloque se calculará como se ha hecho anteriormente. Hay que tener presente que o bien se trabaja con el número de sectores físicos (que serán mayores de 128 bytes) o con sectores de 128 bytes cuando se calcula la capacidad de almacenamiento de cada disco lógico.

El número de entradas del directorio (menos 1) óptimo es de 511 para discos lógicos de 1 megabyte y de 1.023 ó 2.047 para discos mayores. Recuérdese que con la versión 2 de CP/M no se puede tener un disco lógico mayor de 8 megabytes.

Los bloques para el directorio se calculan también como se ha indicado para los discos.

En general, el tamaño de la memoria intermedia del directorio (WADC) se pondrá a 0, ya que no hay necesidad de usar esta característica en sistemas de discos rígidos estancos.

El número de pistas antes del directorio puede utilizarse para dividir el disco físico en discos lógicos más pequeños, como se muestra en la figura 3-9.

No existen reglas que indiquen que las pistas anteriores al directorio del disco lógico no puedan ser utilizadas para contener otros discos lógicos completos. Esto puede verse en la figura 3-9, CP/M se comporta como si cada disco lógico empezase en la pista 0 (y naturalmente es así), pero al

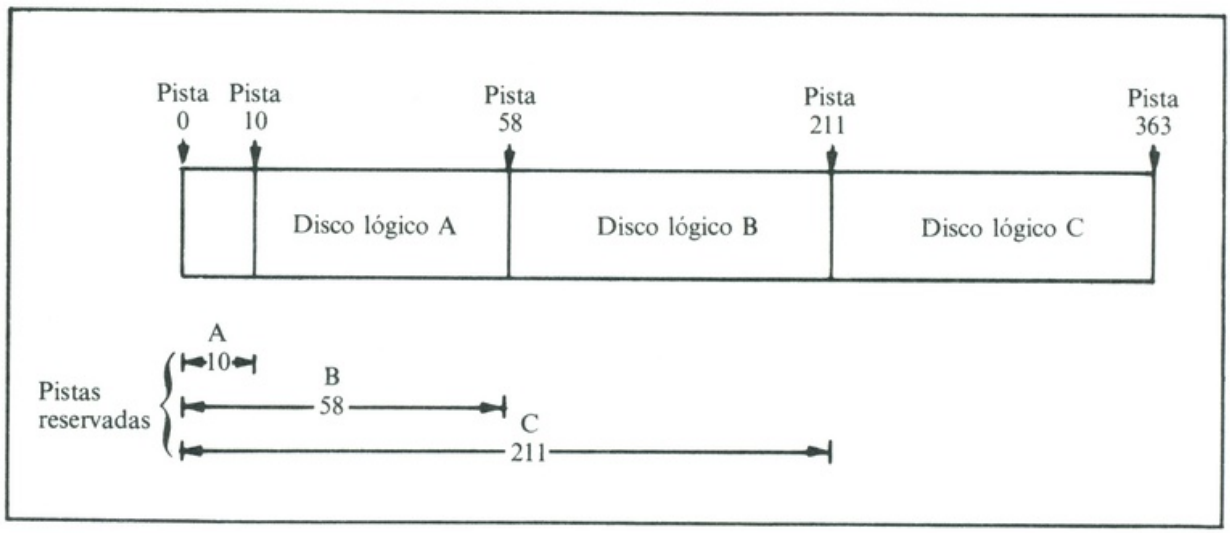


Figura 3-9. División de discos rígidos en discos lógicos.

Características físicas:

364	Pistas/disco
20	Sectores/pista
512	Bytes/sector
10.240	Bytes/pista

Características lógicas elegidas:

	Pistas	Tamaño de bloque
Area reservada	10	n/a
Disco A:	48	2048
Disco B:	153	4096
Disco C:	153	4096

Calcular:

A:	B: y C:	
48	153	Pistas asignadas al disco
× 10.240	× 10.240	Bytes/pista
491.520	1.566.720	Bytes/disco
÷ 2048	÷ 4096	Bytes/bloque
240	382,5	Número de bloques
239	381	Número de bloque máximo

Figura 3-10. Cálculo del número de bloque máximo para un disco rígido.

aumentar el número de pistas antes de cada directorio, los discos lógicos pueden alternarse a través del espacio disponible en el disco físico.

La figura 3-10 muestra los cálculos relacionados con la primera fase de la construcción de los bloques de parámetros del disco para el disco rígido presentado en la figura 3-9. Las características materiales son las impuestas por el diseño del disco rígido. El programador no tendrá ningún control sobre ellas, pero podrá elegir, sin embargo, cuánto disco físico se asigna a cada disco lógico, el tamaño del bloque y el número de entradas del directorio. Como puede verse, el disco lógico A es mucho menor que los discos B y C y estos últimos son del mismo tamaño. El disco A será el disco de sistema desde el que se cargarán la mayoría de los programas, por lo que el tamaño menor de su directorio hará que el cargar los programas sea mucho más rápido. El tamaño del bloque para el disco A es también menor, con el fin de reducir la cantidad de espacio desperdiciado en bloques parcialmente ocupados.

La figura 3-10 muestra también los cálculos relacionados con la búsqueda del número máximo de bloques. Nótese nuevamente que, una vez calculado el número total de bloques, es necesario redondear por defecto, en el caso de que algunos de los componentes sean fraccionarios, y restar luego 1 para obtener el número máximo (siendo 0 el primer bloque).

La figura 3-11 muestra los valores prácticos que se situarán en los bloques de parámetros. Se supone que el controlador de disco es uno del tipo de los que visualizan el disco físico como una serie de sectores contiguos sin hacer referencia a las pistas; la electrónica interna y el "firmware" del controlador cuidan de estos detalles. Por esta razón, se dice que cada sector material es una "pista" en términos del CP/M. Cada "pista" tiene 512 bytes y, por tanto, puede almacenar cuatro sectores de 128

```
DPBA:       DPBB:       DPBC:
      4           4           4          ;Sectores de 128-bytes/pista
      4           5           5          ;Desplazamiento de bloques
     15          31          31          ;Máscara de bloques
      1           1           1          ;Máscara de extensión
    239         381         381          ;Máximo de bloques #
    255        1023        1023          ;Número de entradas al directorio
11110000B   11111111B   11111111B        ;Mapa de bits de los bloques usados
00000000B   00000000B   00000000B        ; por el directorio
      0           0           0          ;Número de bytes en la memoria intermedia de comprobación del directorio
   (10)        (58)       (211)         ;Pistas reales antes del directorio
    200        1160        4220          ;"Pistas" antes del directorio
```

Figura 3-11. Tablas de parámetros de disco para un disco rígido.

bytes. Como puede verse, este es el valor que se encuentra en el campo sectores/"pista".

Los valores de desplazamiento y enmascaramiento de bloque se obtienen de la tabla 3-1, utilizando el tamaño de bloque elegido previamente. Entonces, con el tamaño del bloque y el número máximo de bloques (véase figura 3-10), la máscara de extensión puede obtenerse de la tabla 3-2. En la figura 3-11 puede verse que los valores de la máscara de extensión de 1 fueron obtenidos de la misma manera para los tres discos lógicos, aunque se han elegido dos tamaños distintos de bloque, y aunque el disco A tiene menos de 256 bloques y los discos B y C tienen más.

El mapa de bits que muestra los bloques que se requieren para mantener el directorio de ficheros se calculan multiplicando el número de entradas al directorio por 32 y dividiendo el producto por el tamaño del bloque. El resultado es de 4 para el disco A y de 8 para los discos B y C. Como puede verse, el mapa de bits tiene fijado el número apropiado de bits.

Comoquiera que la mayoría de los discos rígidos que existen hoy en día en el mercado no son transportables, la longitud de la memoria intermedia para control del directorio se pone a cero.

El número de "pistas" antes del directorio requiere un toque de gracia final. Habiendo indicado ya al CP/M que cada "pista" tiene cuatro sectores, es preciso continuar de la misma forma y expresar el número de pistas reales antes del directorio en unidades de sectores físicos de 512 bytes.

Como nota final, si se especifican estos bloques de parámetros para un controlador de disco que requiere que se comunique con él en términos de pistas físicas y sectores de 128 bytes, entonces el número de sectores por pista debe colocarse en 80 (veinte sectores de 512 bytes por pista física). Por tanto, habrá que cambiar el número de pistas antes del directorio expresando el número de pistas físicas (mostrado entre paréntesis en la figura 3-11).

Adición de nueva información al bloque de parámetros

Normalmente, debe asociarse alguna información adicional con todo disco lógico. Por ejemplo, en un sistema que tenga varios discos físicos, es

preciso identificar dónde reside cada disco *lógico*.También puede desearse identificar cualquier otro parámetro físico como tipo de unidad de disco, números de puertos de entrada/salida y dirección de las subrutinas de control.

Puede surgir la tentación de extender la cabecera de parámetros del disco al existir una entrada separada para cada disco lógico. Pero la cabecera de parámetros del disco es exactamente de 16 bytes de longitud; añadiendo más bytes se hace más difícil la aritmética necesaria a utilizar en el BIOS. El mejor lugar para colocar este tipo de información es *prefijarlo* en el principio de cada bloque de parámetros de disco. La etiqueta al principio del bloque debe dejarse en el mismo sitio para que el CP/M no se confunda. Unicamente un programa adicional especial escrito por el programador será lo suficientemente "astuto" como para mirar *delante* del bloque con el fin de encontrar esta información adicional.

Organizaciones de ficheros

El CP/M soporta dos tipos de ficheros secuenciales y de acceso directo o aleatorios. El CP/M visualiza ambos tipos como constituidos por una serie de registros de 128 bytes. Obsérvese que en términos del CP/M un registro es lo mismo que un sector de 128 bytes. Esta terminología a veces puede dificultar la comprensión. Puede ser una ayuda el pensar en los sectores de 128 bytes como registros *materiales*. Los programas de aplicación manipulan registros *lógicos* que poseen poca o ninguna relación con estos registros físicos. Existe código en los programas de aplicación para manipular registros lógicos.

El CP/M no impone restricciones a los contenidos de un fichero. En muchos casos, sin embargo, se utilizan ciertos convencionalismos cuando se almacenan datos de texto. Cada línea de texto se acaba con los caracteres ASCII CARRIAGE RETURN (retorno de carro) y LINE FEED (salto de línea. El último sector de un fichero de texto se llena con caracteres ASCII SUB, 1AH en hexadecimal.

Bloques de control de fichero

Con el fin de conseguir que CP/M trabaje sobre un fichero se necesita procurar una estructura en la cual tanto el programador como el BDOS puedan mantener detalles relevantes sobre el fichero, su nombre y tipo, etc. El bloque de control de fichero (FCB) es una derivación de la entrada del directorio de ficheros, como puede verse en la figura 3-12. Esta figura muestra tanto una serie de igualdades, que pueden utilizarse para acceder a

una entrada, como una serie de instrucciones DB (definición de byte) para presentar un ejemplo.

La primera diferencia que puede apreciarse entre la entrada del directorio de ficheros y el FCB es que el primer byte sirve para un propósito diferente. En el FCB se utiliza para especificar en qué disco se encuentra el fichero. Recordemos que en el directorio este byte indica el número de usuario para una entrada determinada. Cuando se procesan prácticamente los ficheros, el número de usuario en curso se fija bien por el operador en una orden de consola o bien mediante una llamada a función del BDOS; esto predefine qué subconjunto de ficheros del directorio será procesado. Por consiguiente, el FCB no necesita mantener información del número de usuario.

El número de disco está almacenado en el primer byte del FCB de forma rara. Un valor 0 indica al CP/M que tiene que buscar el fichero en el disco por defecto en curso. Este disco por defecto se selecciona bien por medio de una entrada desde la consola o bien haciendo una llamada específica al BDOS dentro de un programa. En general, el disco por defecto debe prefijarse en el disco que contiene el conjunto de programas con los que se esté trabajando, lo que evita mecanografiado innecesario en el teclado cuando se quiere cargar un programa.

Un valor de disco distinto de 0 representa una letra del alfabeto, basado en un esquema de codificación sencillo de A=1, B=2, etc.

Como puede verse en la figura 3-12, el nombre y tipo de fichero deben

```
0000 =          FCBE$DISK       EQU     0       ;Unidad de disco (0 = por defecto, 1=A)
0001 =          FCBE$NAME       EQU     1       ;Nombre de fichero
0009 =          FCBE$TYP        EQU     9       ;Tipo de fichero
                                                ;Sobrante de bits usados para tipo
0009 =          FCBE$RO         EQU     9       ;Bit 7 = 1 - sólo lectura
000A =          FCBE$SYS        EQU     10      ;Bit 7 = 1 - modo sistema
000B =          FCBE$CHANGE     EQU     11      ;Bit 7 = 0 - fichero escrito
                                                ;
000C =          FCBE$EXTENT     EQU     12      ;Número de extensión
                                                ;13, 14 reservado para CP/M
000F =          FCBE$RECUSED    EQU     15      ;Registros utilizados en esta extensión
0010 =          FCBE$ABUSED     EQU     16      ;Bloques utilizados
0020 =          FCBE$SEQREC     EQU     32      ;Registro secuencial para lectura/escritura
0021 =          FCBE$RANREC     EQU     33      ;Registro de acceso directo para lectura/escritura
0023 =          FCBE$RANRECO    EQU     35      ;Byte de desbordamiento de registro secuencial
                ;
                ;
                ;
0000 00         FCB$DISK:       DB      0       ;Búsqueda en unidad de disco por defecto
0001 46494C454EFCB$NAME:        DB      'FILENAME'      ;Nombre de fichero
0009 545950     FCB$TYP:        DB      'TYP'   ;Tipo de fichero
000C 00         FCB$EXTENT:     DB      0       ;Extensión
000D 0000       FCB$RESV:       DB      0,0     ;Reservado para CP/M
000F 00         FCB$RECUSED:    DB      0       ;Registros utilizados en esta extensión
0010 0000000000FCB$ABUSED:      DB      0,0,0,0,0,0,0,0   ;Bloques utilizados
0018 0000000000                 DB      0,0,0,0,0,0,0,0
0020 00         FCB$SEQREC:     DB      0       ;Registro secuencial para lectura/escritura
0021 0000       FCB$RANREC:     DW      0       ;Registro de acceso directo para lectura/escritura
0023 00         FCB$RANRECO:    DB      0       ;Byte de desbordamiento de registro secuencial
```

Figura 3-12. Declaraciones de datos para el FCB.

ser prefijados en los valores requeridos y, para el proceso de ficheros secuenciales, el resto del FCB puede ponerse a cero. Hablando estrictamente, los últimos tres bytes del FCB (el número de registro de acceso aleatorio y el byte de desbordamiento de registro de acceso aleatorio) tampoco necesitan ser declarados si no se va a procesar nunca el fichero en acceso directo.

Esto conduce a un punto conceptual sutil. Los ficheros de acceso directo lo son únicamente porque *el programador* los procesa de esta forma. Aunque esto suena a perogrullada, significa que los ficheros de CP/M no son intrínsecamente ni de acceso aleatorio ni secuenciales. Lo que sean depende de la elección del programador al procesarlos en cada punto determinado. Por consiguiente, comoquiera que la forma en que se procesan será diferente, no existe ningún aditamento especial en el fichero que indique cómo ha de utilizarse.

Ficheros secuenciales

Un fichero secuencial comienza al principio y acaba al final. Puede verse como una sucesión de registros de 128 bytes.

Para crear un fichero secuencial, hay que declarar un bloque de control de fichero con el nombre y tipo de fichero requeridos y solicitar al BDOS que lo *cree*. Entonces puede solicitarse al BDOS que escriba "registro" por "registro" (realmente sector de 128 bytes por sector de 128 bytes) en él. El BDOS se ocupará de abrir nuevas extensiones si es necesario. Cuando se han escrito todos los datos, hay que solicitar al BDOS que cierre el fichero.

Para leer un fichero existente se necesitará también un FCB con el nombre y tipo de fichero requerido ya declarado. Entonces se requiere al BDOS para que abra el fichero para el procesamiento y una serie de peticiones de lectura secuencial, cada una de las cuales lleva al próximo registro hasta que o bien el programa detecta una condición de final de fichero (mediante el examen de los datos que se reciben del mismo) o bien el BDOS descubra que no existen más sectores que leer en él. No hay necesidad de cerrar un fichero del que se han estado leyendo datos —pero *es preferible cerrarlo*—. Esto no es necesario si se va a hacer funcionar el programa únicamente con el CP/M, pero sí es necesario si se desea trabajar con MP/M (la versión para múltiples usuarios del CP/M).

¿Qué ocurre si se necesita añadir más información a un fichero existente? Una opción es crear un nuevo fichero, copiar el existente en el nuevo y empezar entonces a añadir datos al final del nuevo fichero. Afortunadamente, con CP/M esto no es necesario. En el FCB que se usa para leer un fichero, el nombre y tipo estaban especificados, pero puede especificarse también el número de extensión. Si se hace así, el BDOS procederá a abrir (si lo encuentra) el número de extensión que se le solicita. Si el BDOS abre

la extensión satisfactoriamente, todo lo que hace falta es comprobar si el número de registros utilizado en la extensión (contenido en el campo FCB$RECUSED) es menor de 128 (80H). Esto indica que la extensión no está llena. Tomando este número y colocándolo en el byte FCB$SEQREC (número de registro secuencial) del FCB, puede hacerse que CP/M *salte hacia delante* y empiece a escribir desde el final efectivo del fichero.

Ficheros de acceso aleatorio (acceso directo)

Los ficheros de acceso aleatorio utilizan una sencilla variación de la técnica que acabamos de describir. La principal diferencia es que el número de registro ha de ser prefijado en el FCB. El BDOS mantiene automáticamente la información de las extensiones de fichero durante las peticiones de lectura/escritura aleatoria. (Estas peticiones se explican más ampliamente en el capítulo 5.)

Conceptualmente, los ficheros aleatorios necesitan un pequeño viraje del pensamiento. Después de crear un fichero en la forma descrita anteriormente, hay que prefijar el número de registro en el FCB antes de solicitar cualquier escritura. Esto es un valor de dos bytes denominado FCB$RANREC en la figura 3-12. Entonces, cuando se da la instrucción de escritura aleatoria al BDOS, éste mirará el número de registro, calcula en qué extensión debe existir el registro; si es necesario, crea la entrada de directorio para la extensión, y, finalmente, escribe los datos. Utilizando este esquema, se puede avanzar y retroceder rápidamente en el fichero colocando registros al azar en todo el espacio de fichero, con CP/M creando las entradas de directorio necesarias cada vez que se aventura en una parte del fichero que aún no ha sido escrita.

La misma técnica se utiliza para leer un fichero de acceso directo. Se fija el número de registro en el FCB y entonces se realiza una llamada al BDOS para que abra la extensión correcta y lea los datos. El BDOS devolverá un error si no puede encontrar la extensión requerida o si este registro particular no existe.

Para el irreflexivo existen problemas de espera. Antes de empezar a leer o escribir, es preciso abrir el fichero en la extensión 0, aun cuando esta extensión puede no contener ningún registro de datos. Para un nuevo fichero, esto puede hacerse con la petición de creación de fichero, y para uno existente con la petición normal de apertura de fichero. Si se crea un fichero *esparcido,* que presente huecos entre los datos, pueden surgir problemas al manipularlo. Ocurrirá que tiene varias extensiones, cada una de ellas parcialmente llenas, lo que engañará a algunos programas que normalmente procesan ficheros secuenciales, al no esperar ver una extensión parcialmente llena, excepto al final de un fichero, y puede colocarse al final en un sitio equivocado.

Funciones del CCP
Edición de líneas de órdenes del CCP
Ordenes incorporadas o permanentes
Carga de programas
Página base
Descarga de memoria de la página base
Procesamiento de la parte de parámetros de orden
Memoria disponible
Comunicación con el BIOS
Regreso al CP/M

4

El procesador de órdenes de consola (CCP)

El procesador de órdenes de consola procesa las órdenes que se introducen desde la consola. Como se recordará de la breve introducción hecha en el capítulo 2, el CCP se carga en la memoria inmediatamente debajo del BDOS. En la práctica, muchos programas escriben deliberadamente encima del CCP, a fin de usar la memoria que ocupa normalmente. Esto proporciona al programa una cantidad adicional de 800H bytes (2K bytes).

Cuando uno de estos "programas transitorios" termina, cede el control al BIOS, que a su vez vuelve a cargar una nueva copia del CCP desde las pistas del sistema del disco a la memoria y entonces le transfiere el control. Por consiguiente, el CCP tiene una existencia esporádica —una serie interminable de cargas en la memoria, aceptando una orden del programador en la consola, escribiéndose sobre él el programa que se solicita sea

cargado y siendo llevado otra vez a la memoria cuando el programa termina.

Este capítulo trata acerca de lo que el CCP realiza en estos breves períodos en que se encuentra en memoria.

Funciones del CCP

Expresado sencillamente, una vez que el CCP posee el control de la máquina, también lo posee el programador. El CCP anuncia su presencia mostrando un aviso de dos caracteres, una letra del alfabeto para indicar el disco en curso y un signo "mayor que". En el ejemplo A>, la "A" significa que el disco en curso por defecto es A, y el ">", que el mensaje ha sido producido por el CCP.

Una vez visto el aviso, el CCP está preparado para que el programador introduzca una línea de órdenes. Una línea de órdenes consiste en dos partes principales: el nombre de la orden y, opcionalmente, algunos valores para la orden. Esta última parte se conoce con el nombre de *parámetros de la orden o cola de la orden.*

La orden en sí misma puede ser una de estas dos cosas: bien un nombre de un fichero o bien el nombre de una de las órdenes incorporadas al CCP, de uso frecuente.

Al introducir el nombre de una de las órdenes incorporadas al CCP, éste no necesita ir al disco para cargar el programa para ejecución. El código ejecutable está ya en el CCP.

Si el nombre de la orden no corresponde a ninguna de las permanentes (éste posee una tabla con dichos nombres), el CCP elegirá la unidad de disco lógico adecuada para un fichero con este nombre y tipo "COM" (abreviatura de comando, orden). No se introduce COM cuando se está llamando a una orden —el CCP supone "COM" como tipo de fichero.

Si el nombre del fichero COM no va precedido de una especificación de la unidad de disco lógico, el CCP buscará en la unidad de disco por defecto en curso. Si el nombre del fichero COM va precedido por el de una unidad lógica específica, el CCP solamente buscará el programa en esta unidad. Por ejemplo, la orden MYPROG provocará que el CCP busque un fichero denominado "MYPROG.COM" en la unidad de disco por defecto, mientras que C:MYPROG hará que el CCP busque únicamente en la unidad C.

Si se introduce el nombre de una orden que no coincide ni con la tabla de órdenes incorporadas al CCP ni con el nombre de ningún fichero ".COM" en el disco especificado, el CCP sacará el nombre de la orden seguido de un signo de interrogación, indicando así que no es capaz de encontrar el fichero.

Edición de líneas de órdenes del CCP

El CCP utiliza una memoria intermedia de una línea para almacenar lo que se mecanografía hasta que se pulsa o bien CARRIAGE RETURN o LINE FEED. Si se comete un error o se cambia de opinión, se puede modificar una orden incompleta, volviendo hasta el punto de error.

Se edita una línea de órdenes introduciendo *caracteres de control* desde la consola. Los caracteres de control se designan bien por la combinación de claves requeridas para generarlas desde el teclado o bien por su nombre oficial en el conjunto de caracteres ASCII. Por ejemplo, CONTROL-J se conoce también como CARRIAGE RETURN o CR.

Cuando el CP/M ha de representar caracteres de control, lo convencional es indicar el aspecto de "control" de un carácter con un signo "^". Por ejemplo, CONTROL-A aparecerá como "^A", CONTROL-Z como "^Z", etc. Pero si se pulsa la tecla de control con la de cambio a "mayúsculas" y la "6", se producirá un CONTROL-^ o "^^". La representación de claves de control con este signo solamente es necesaria cuando se extraen estos caracteres para la consola o la impresora —interiormente, se mantienen en sus valores binarios adecuados.

CONTROL-C: Arranque en caliente Si se introduce un CONTROL-C como primer carácter de una línea de órdenes, el CCP iniciará una operación de arranque caliente. Esta operación reinicializa el CP/M completamente, incluido el sistema de disco. Se carga una copia nueva del CCP en la memoria y se relee el directorio de ficheros de la unidad de disco por defecto en curso, reconstruyéndose el mapa de bits de situación que contiene el BIOS (como se indicó en el capítulo 3).

La única ocasión en que se iniciará una operación de arranque caliente es después de haber cambiado un disquete (o un disco, si se tienen discos duros transportables). Entonces el CP/M reinicializará el sistema de disco.

Obsérvese que un CONTROL-C inicia un arranque caliente sólo si éste es el primer carácter de una línea de órdenes. Si se introduce en otra posición cualquiera, el CCP únicamente lo mostrará en la pantalla como "^C". Si se han introducido ya algunos caracteres en una línea de órdenes, úsese CONTROL-U o CONTROL-X para cancelar la línea, y entonces CONTROL-C para iniciar un arranque caliente. Se nota que se ha realizado un arranque caliente al existir una pausa apreciable después del CONTROL-C antes de que aparezca el próximo aviso. El sistema necesita un tiempo finito para releer el directorio de ficheros y reconstruir el mapa de bits de situación.

CONTROL-E: Final de línea física La orden de CONTROL-E es una reliquia de los días del teletipo y los terminales que no realizan un retorno automático de carro y salto de línea cuando el cursor sale de la pantalla por la derecha. Cuando se mecanografía un CONTROL-E, el CP/M envía una orden de

CARRIAGE RETURN/LINE FEED a la consola, pero no comienza a ejecutar la línea de orden que se ha mecanografiado hasta el momento. El CONTROL-E es, efectivamente, un final de línea *física* y no *lógica*.

Como puede verse, se necesitará usar esta orden sólo si el terminal escribe encima de la propia línea (si se trata de un mecanismo de copia) o si no retorna el cursor cuando va a la derecha del final de la línea.

CONTROL-H: Retroceso CONTROL-H es el carácter ASCII de retroceso. Cuando se mecanografía, el CCP mueve el cursor hacia atrás "destructivamente". Debe utilizarse para corregir errores mecanográficos que se descubren antes de introducir la línea de órdenes. El último carácter mecanografiado desaparecerá de la pantalla. El CCP realiza este trabajo enviando una secuencia de tres caracteres: retroceso, espacio, retroceso a la consola.

El CCP ignora los intentos de retroceso sobre sus propios mensajes. También se ocupa del retroceso en el caso de los caracteres de control que ocupan dos posiciones de carácter en la línea. El CCP envía la de secuencia de caracteres retroceso, retroceso, espacio, espacio, retroceso, retroceso, borrando ambos caracteres.

CONTROL-J: Salto de línea/CONTROL-M: Retorno de carro CONTROL-J es el carácter LINE FEED del ASCII, el CONTROL-M es el CARRIAGE RETURN. Ambos terminan la línea de orden. El CCP ejecutará entonces la orden.

CONTROL-P: Eco de impresora La orden CONTROL-P se utiliza para conectar y desconectar un dispositivo denominado *eco de impresora*. Cuando está conectado, todos los caracteres enviados a la consola se envían también al mecanismo de listado del CP/M. Puede usarse este mando para obtener una copia en papel, por ejemplo de la información que normalmente va sólo a la consola.

El CONTROL-P es un "conmutador". La primera vez que se mecanografía CONTROL-P conecta el eco de impresora; la próxima vez que se escribe CONTROL-P, lo desconecta. Siempre que el CP/M realiza un arranque caliente, el eco de impresora queda desconectado.

No existe un sistema fácil para saber si el eco de impresora está conectado o desconectado. Puede intentarse mecanografiar algunos CARRIAGE RETURNs, y ver si la impresora responde; si no lo hace, hay que mecanografiar CONTROL-P y volver a intentarlo.

Uno de los defectos en la mayoría de las implementaciones del CP/M es que los controladores de la impresora (el software del BIOS que controla o "impulsa" la impresora) no se comportan de forma muy inteligente si la impresora está desconectada o no preparada cuando el programador o el programa le piden que imprima. Bajo estas circunstancias, el software esperará eternamente y el sistema quedará como muerto. Por consiguiente, cuando se cuelgue el sistema de esta forma al mecanografiar un CONTROL-P,

hay que controlar que la impresora está conectada y dispuesta. De otra forma, quizá haya que inicializar de nuevo todo el sistema.

CONTROL-R: Repetir línea de órdenes La orden CONTROL-R hace que el CCP repita o vuelva a escribir la línea en curso. El CCP emite un carácter "#", un CARRIAGE RETURN/LINE FEED, y después todo el contenido de la memoria interna de línea de órdenes. Este dispositivo es muy útil si se está trabajando en un teletipo u otro terminal de copia dura y si se han usado caracteres RUB o DEL. Como estos caracteres no borran de forma destructiva un carácter, se puede obtener una línea visualmente confusa del texto en el terminal. El carácter CONTROL-R proporciona una nueva copia de la línea sin ninguno de los caracteres borrados lógicamente. De esta forma se puede ver exactamente lo que está mecanografiado en la memoria interna de línea de órdenes.

Véase lo que se trata acerca de los caracteres RUB y DEL como ejemplo de CONTROL-R.

CONTROL-S: Detención de salida por pantalla La orden CONTROL-S es el carácter XOFF del ASCII (llamado también DC3); XOFF es una abreviatura de "Transmit Off". Mecanografiando CONTROL-S se para temporalmente la salida a la consola. En una versión estándar del CP/M, el CCP reanudará la salida cuando entre algún carácter (incluido otro CONTROL-S) desde la consola. Por tanto, se puede usar CONTROL-S como un interruptor para conectar y desconectar la salida por la consola.

En algunas implementaciones del CP/M, el propio controlador de la consola (código de bajo nivel del BIOS que controla la consola) mantendrá un protocolo de comunicación con la consola; por tanto, el modo más adecuado para reanudar la salida por la consola después de una pausa debida a un CONTROL-S es utilizar CONTROL-Q, el XON del ASCII, o carácter de "transmisión". La introducción de un CONTROL-Q en lugar de fiarse del hecho de que *todos* los caracteres pueden usarse para continuar la salida es una forma de evitar fracasos.

Las órdenes CONTROL-S y CONTROL-Q son las más útiles cuando se tienen grandes cantidades de datos en la pantalla. A base de utilizar los controles S y Q, puede dejarse que los datos lleguen a la pantalla en pequeñas fracciones que pueden leerse fácilmente.

CONTROL-U o CONTROL-X: Anulación de la línea de orden Las órdenes CONTROL-U y CONTROL-X realizan la misma función: Borran la línea de órdenes en curso parcialmente introducida, de forma que se pueden anular todos los errores y empezar otra vez. La orden CONTROL-U estaba destinada originalmente a terminales de copias duras. El CCP muestra un carácter "#", después un CARRIAGE RETURN/LINE FEED y luego algunos espacios para dejar el cursor alineado y dispuesto para que se introduzca la próxima

línea de órdenes, y deja lo que se había introducido originalmente en la línea anterior de la pantalla. CONTROL-X es más propio para las pantallas; el CCP retrocede destructivamente al principio de la línea de órdenes, de forma que ésta pueda reintroducirse.

RUB o DEL: Borrado del último carácter La función de borrado (teclas marcadas RUB, RUBOUT, DEL o DELETE) borran de forma no destructiva el último carácter mecanografiado. Es decir, borra el último carácter en la memoria de líneas de órdenes y lo repite en la consola.

He aquí un ejemplo de una línea de órdenes con los últimos caracteres borrados utilizando la clave RUB:

```
A>RUN PAYROLLLLORYAPSALES
              ^^^^^^^
              DELeted
```

Como puede verse, la línea de órdenes se hace rápidamente ilegible. Si se pierde la idea de lo que son caracteres de datos y de lo que ha sido borrado, puede usarse CONTROL-R para obtener una nueva copia de lo que está en la memoria de líneas de órdenes.

El ejemplo anterior aparecerá entonces como sigue:

```
A>RUN PAYROLLLLORYAPSALES#
  RUN SALES_
```

El carácter "#" se muestra por el CCP para indicar que la línea ha sido repetida. El "_" representa la posición del cursor, que está ahora dispuesto para continuar con la línea de órdenes.

Ordenes incorporadas o permanentes

Cuando se introduce una línea de órdenes y se pulsa CARRIAGE RETURN o LINE FEED, el CCP controlará si el nombre de la orden es uno del conjunto de las incorporadas. (Posee una pequeña tabla de nombres de él, con la cual se contrasta el nombre de la orden introducida.) Si el nombre de la orden introducida concuerda con uno de los incorporados, el CCP ejecuta la orden inmediatamente.

Las próximas secciones describen las órdenes incorporadas disponibles; sin embargo, para obtener una información más amplia con ejemplos de las distintas formas de cada una de ellas, consúltese la *Guía de Usuarios del CP/M Osborne,* segunda edición, de Thom Hogan (Berkeley: Osborne/McGraw-Hill, 1982).

X:—Cambio de la unidad de disco implícita La unidad implícita o "por defecto" es la normalmente activa que el CP/M utiliza para acceder a todos los ficheros, siempre que no se mencione una unidad específica. Si se desea cambiar la unidad por defecto, introdúzcase simplemente la letra de identificación de la nueva unidad que se tomará por defecto seguida de dos puntos. El CCP responde cambiando el nombre del disco que aparece en la línea de mensaje.

En los discos rígidos esta simple operación puede necesitar uno o dos segundos para completarse, porque el BDOS, al ser requerido por el CCP para que se introduzca en la unidad, debe leer el directorio del disco y reconstruir el vector de situación para él. Si se tiene un disquete o un disco transportable, cambiándolo y realizando un arranque en caliente se obtiene el mismo efecto que renovando la imagen que tiene CP/M de los bloques que se han usado y los que hay disponibles. En un disco rígido se tarda mucho más, porque, en general, los directorios son mucho mayores.

DIR—Directorio de ficheros En su forma más sencilla, la orden DIR presenta una lista de los ficheros situados a modo de directorio con el número de usuario en curso (o de grupo de ficheros) y en la unidad por defecto en curso. Por tanto, cuando no se solicita ningún fichero después de la orden DIR, se supone un nombre de fichero "*.*". Esto es un "comodín" total, por lo que aparecerán todos los ficheros que no se hayan puesto en modo sistema. Esta es la única orden permanente que actúa de forma que un nombre de fichero omitido se entiende como "todos los nombres de fichero y todos los tipos de fichero".

Se puede llamar al directorio de una unidad distinta especificando la unidad en la misma línea que la de la orden DIR.

Se pueden calificar los ficheros que se desee llamar introduciendo un nombre de fichero o extensión único o ambiguo. Aparecerán en pantalla solamente aquellos ficheros que correspondan a la especificación dada y, aún así, sólo aquellos que no están puestos en modo sistema aparecerán en la pantalla. (El programa de utilidad estándar del CP/M STAT puede utilizarse para cambiar ficheros de modo SYS a modo DIR.)

Otro efecto secundario de la orden DIR y de ficheros que están en modo SYS se ilustrará mejor con un ejemplo. Imagínese que la unidad lógica en curso B contiene dos ficheros, denominados SYSFILE (que está en modo SYS) y NONSYS (que no lo está). Obsérvese el siguiente diálogo de consola, en el cual las entradas que corresponden al operador están subrayadas:

```
B>DIR<cr>
B: NONSYS                no aparece el archivo SYS
B>DIR JUNK<cr>
NO FILE                  no existe el JUNK
B>DIR SYSFILE<cr>
B>_
```

¿Comprenden el problema? Si un fichero no está en el disco, el CCP hará aparecer NO FILE (o NOT FOUND en versiones anteriores del CP/M). Sin embargo, si el fichero *existe*, pero es un fichero en modo SYS, el CCP no lo presentará a causa de su situación; pero tampoco indicará NO FILE. En lugar de ello reinserta tranquilamente la indicación. Esto puede inducir a confusión si se está buscando un fichero en modo SYS. El único camino seguro que existe para saber si existe el fichero es utilizar STAT.

ERA—Borrado de un fichero La orden ERA borra "lógicamente" ficheros del disco (*lógicamente* porque sólo es afectado el directorio, los bloques de datos no cambian).

El borrado lógico cambia el primer byte de cada entrada al directorio que pertenezca al fichero por el valor 0E5H. Como puede recordarse de lo tratado sobre la entrada al directorio en el capítulo 3, este primer byte contiene usualmente el número de usuario. Si se pone en 0E5H, marca la entrada como si hubiera sido borrada.

ERA realiza un repaso completo del directorio de ficheros para borrar lógicamente todas las extensiones del mismo.

Al contrario de DIR, la orden ERA no supone "todos los ficheros, todos los tipos" si se omite el nombre de un fichero concreto. Si esto ocurriera sería demasiado fácil borrar todos los ficheros por accidente. Hay que introducir "*.*" para borrar todos los ficheros y, aún así, hay que confirmar al CCP que realmente se desea borrarlos todos del disco. El diálogo práctico se presenta de la siguiente forma:

```
A>era b:*.*<cr>
ALL (Y/N)?y<cr>
A>_
```

Si se cambia de opinión en el último instante, puede pulsarse "n" y el CCP no borrará ningún fichero.

Un defecto del CCP es que la orden ERA sólo pide confirmación cuando se intenta borrar todos los ficheros utilizando un nombre tal como "*.*" o "*.???". Considérese el impacto de la orden siguiente:

```
A>ERA *.C??<cr>
A>_
```

El CCP borra sin vacilar todos los ficheros que tienen un campo tipo de fichero que empiece por la letra "C" en el número de usuario en curso del disco A.

Si se necesita usar un nombre de fichero ambiguo en una orden ERA, compruébese qué ficheros se borrarán utilizando antes una orden STAT, con el mismo nombre de fichero ambiguo exactamente. El STAT mostrará todos los ficheros que concuerden con el nombre ambiguo, incluso aquellos que están en modo SYS que no aparecerían tras una orden DIR.

Existen varios programas de utilidad en el mercado con nombres como UNERA o WHOOPS, que toman un nombre de fichero ambiguo y vuelven a crear los ficheros que hayan podido borrarse accidentalmente. En el capítulo 11 se trata acerca de una versión del UNERASE.

Si se intenta borrar un fichero que no se encuentra en la unidad específica, el CCP responderá con un mensaje de NO FILE.

REN—Renombrar un fichero La orden REN da nuevo nombre a un fichero, cambiando su nombre, su tipo o ambos. Para cambiar de nombre se necesita introducir los dos nombres del fichero, el nuevo y el actual.

Para recordar el formato correcto, piénsese en la frase *nuevo=viejo* (new=old). La sintaxis de la orden es:

```
A>ren newfile.typ=oldfile.typ<cr>
A>_
```

Se puede usar la letra de la unidad de disco lógico para especificar en qué unidad existe el fichero. Si se especifica la unidad, sólo es necesario introducirla en uno de los nombres de fichero. Si se introduce la unidad en ambos nombres de fichero, deberá ser con la misma letra para ambos.

A diferencia de las órdenes permanentes anteriores, REN no puede utilizarse con nombres de fichero ambiguos. Si se intenta, el CCP devuelve los nombres ambiguos con un signo de interrogación, como se indica en el diálogo siguiente:

```
A>ren chap*.doc=chapter*.doc<cr>
CHAP*.DOC=CHAPTER*.DOC?
A>_
```

Si la orden REN no puede encontrar el fichero "viejo", responderá NO FILE. Si el fichero nuevo existe ya, el mensaje que aparecerá es FILE EXISTS. Si se recibe un mensaje de FILE EXISTS y se quiere comprobar que el nuevo fichero existe realmente, recuérdese que es mejor usar una orden STAT que una DIR. El fichero existente puede haber sido declarado modo SYS y, por tanto, no aparecerá si se usa la orden DIR.

TYPE—Visualizar un fichero que contiene texto La orden TYPE copia el fichero especificado en la consola. No se pueden utilizar nombres de fichero ambiguos, y se necesitará pulsar CONTROL-S si el fichero tiene más datos de los necesarios para llenar una pantalla. Con la orden TYPE, los datos del fichero pasarán rápidamente por la pantalla a menos que se pulse CONTROL-S para detener la salida. Es preciso tener cuidado, porque si se pulsa cualquier otro carácter, la orden TYPE abortará y devolverá el control al CCP.

Una vez se ha comprobado lo que aparece en la pantalla, puede pulsarse

CONTROL-Q para reanudar la salida de datos a la consola. Con las implementaciones estándar del CP/M se descubrirá que cualquier carácter puede utilizarse para poner en marcha de nuevo la circulación de datos; sin embargo, el CONTROL-S (X-OFF) y CONTROL-Q (X-ON) constituyen el protocolo estándar que ha de utilizarse.

Si se necesita la salida de copias duras de los contenidos del fichero, deberá mecanografiarse una orden CONTROL-P antes de pulsar el retorno de carro (CARRIAGE RETURN) al final de la línea de orden TYPE.

Como se habrá observado, la orden TYPE sólo debe utilizarse para salida de ficheros de texto ASCII. Si por cualquier razón se utiliza la orden TYPE con un fichero que contenga información binaria, aparecerán en la pantalla caracteres extraños. En este caso se puede programar el terminal en algún estado que sólo pueda corregirse desconectando el aparato y volviéndolo a conectar. Por consiguiente, la regla general es utilizar el mando TYPE *sólo* con ficheros de texto ASCII.

SAVE—Guardar en el disco la imagen en memoria La orden SAVE es la más difícil de explicar de las del CCP. Es más útil para el programador que para un usuario final típico. El formato de esta orden es:

```
A>SAVE n FILENAME.TYP<cr>
A>_
```

La orden SAVE crea un fichero de nombre y tipo especificados (o escribe sobre un fichero existente de este nombre y tipo), introduciendo en él el número *n* de páginas de memoria especificado. Una página en CP/M es de 256 (100H) bytes. La orden SAVE comienza a transcribir la memoria desde el lugar 100H, el principio del área de programa transitorio (TPA). Antes de usar esta orden se habrá cargado normalmente un programa en la TPA. La orden SAVE realiza exactamente lo que su nombre indica: salva una imagen del programa en un fichero en disco.

Frecuentemente, cuando se usa la orden SAVE, el tipo de fichero será ".COM". Con el fichero guardado en esta forma, el CCP será capaz de cargar y ejecutar el fichero.

USER—Cambio del número de usuario Como se ha mencionado anteriormente, el directorio de todo disco lógico consiste en varios directorios entrelazados pero separados lógicamente por el número de usuario. Cuando se usa un número específico de usuario, los ficheros que fueron creados cuando se estaba en otro número de usuario no están lógicamente disponibles.

La orden USER es una forma para desplazarse de un número de usuario a otro. El formato de la orden es:

```
A>USER n<cr>
A>_
```

donde *n* puede ser cualquier número entre 0 y 15. Cualquier otro número provocará que el CCP vuelva a repetir la entrada, seguida de un signo interrogativo.

Pero una vez que se ha ido hacia atrás y hacia delante cambiando el número de usuario varias veces, es fácil equivocarse acerca del número de usuario en que se está. La orden STAT puede utilizarse para encontrar el número de usuario en curso. Si se está en un número de usuario que no tiene una copia del STAT disponible, todo lo que se puede hacer es utilizar la orden USER para situarse en otro número de usuario. No se puede saber el número de usuario en el que se estaba; únicamente se puede indicar al sistema el número de usuario al que se desea ir.

En los sistemas BIOS, a medida que se verán más adelante, existe un medio para hacer aparecer el número actual de usuario cada vez que se lleva a cabo un arranque en caliente. Si se está construyendo un sistema en el cual se planifica utilizar las características del número de usuario CP/M, es preciso pensar seriamente en la presentación del número de usuario actual. Si se está en un número de usuario equivocado y se borran ficheros, pueden crearse serios problemas.

Algunas implementaciones del CP/M han modificado el CCP de tal forma que el mensaje muestra el número actual de usuario, así como la unidad por defecto (semejante al mensaje utilizado en el MP/M). Sin embargo, el uso de un CCP no estándar no es una buena práctica. Generalmente, la definición a medida debe reservarse al BIOS.

Carga de programas

El área que se ha de considerar en primer lugar cuando se carga un programa es la de los 100H primeros bytes de memoria, llamados *página base*. Algunos campos —unidades de esta área de memoria— están fijados previamente a determinados valores antes de que el programa tome el control.

Para ayudarnos en este asunto, imagínese un programa denominado COPYFILE que copie un fichero en otro. Este programa espera que se le especifiquen los nombres de la fuente y del fichero de destino en la línea de órdenes. Una orden típica sería:

```
A>copyfile tofile.typ fromfile.typ display
```

Obsérvese la palabra "display". COPYFILE, si se especifica la opción "display", sacará los contenidos del fichero fuente ("fromfile.typ") a la consola mientras tiene lugar la transferencia.

Al pulsar la tecla CARRIAGE RETURN al final de la línea de orden, el

CCP buscará la unidad por defecto en curso ("A" en el ejemplo) y cargará un fichero llamado COPYFILE.COM en la memoria, empezando en la posición 100H. El CCP transfiere entonces el control a la posición 100H —exactamente después de la página base— y comienza a ejecutarse el COPYFILE.

Página base

La página base generalmente comienza desde la posición 0000H de la memoria, pero si hay otras cosas en las direcciones de memoria bajas puede empezar en una dirección más alta. La figura 4-1 muestra el programa en lenguaje ensamblador que puede necesitarse para acceder a la página base. En este ejemplo se supone que la RAM empieza en la posición 0000H.

```
0000 =      RAM             EQU     0       ;Comienzo de la RAM (y página base)
                                            ;Puede ser necesario cambiarla por
                                            ; algún otro valor (por ejemplo, 4300H)

0000        ;               ORG     RAM     ;Fijación del contador de posiciones a la base de la RAM
0000        WARMBOOT:       DS      3       ;Contiene un JMP a la entrada de arranque caliente
                                            ; en la tabla de vectores de salto del BIOS
            ;
0002 =      BIOSPAGE        EQU     RAM+2   ;Página de vectores de salto del BIOS
            ;
0003        IOBYTE:         DS      1       ;Byte de redireccionamiento de entrada/salida
            ;
0004        CURUSER:        DS      1       ;Usuario en curso (bits 7-4)
0004 =      CURDISK         EQU     CURUSER ;Disco lógico por defecto (bits 3-0)
            ;
0005        BDOSE:          DS      3       ;Contiene un salto a la entrada del BDOS
0007 =      TOPRAM          EQU     BDOSE+2 ;Comienzo de página de la RAM utilizable
            ;
005C                        ORG     RAM+5CH ;Salto de posiciones no utilizables
            ;
005C        FCB1:           DS      16      ;Bloque de control de fichero #1
                                            ;Nota: Si se utiliza este FCB aquí, puede
                                            ; escribirse encima del FCB2 que sigue.
            ;
006C        FCB2:           DS      16      ;Bloque de control de fichero #2
                                            ;Puede situarse en otra posición
                                            ; antes de ser usado
            ;
0080                        ORG     RAM+80H ;Salto de posiciones no utilizables
            ;
            COMTAIL:                        ;Inclusión de los parámetros de orden
0080        COMTAIL$COUNT:  DS      1       ;Cuenta del número de caracteres de la parte de parámetros
                                            ; de la orden (no incluye CR)
0081        COMTAIL$CHARS:  DS      127     ;Caracteres de la parte de parámetros
                                            ; de una orden convertidos a mayúsculas y
                                            ; sin tener en cuenta CR.
            ;
0080                        ORG     RAM+80H ;Redefinición del área de parámetros de órdenes
            ;
0080        DMABUFFER:      DS      128     ;Dirección DMA por defecto utilizada
                                            ; como memoria-registro de 128-bytes
            ;
0100                        ORG     RAM+100H ;Salto de posiciones no utilizadas
            TPA:                            ;Comienzo del área de programa transitorio
                                            ; en la que se cargan los programas.
```

Figura 4-1. Página base de declaraciones de datos.

Algunas versiones del CP/M, tales como el sistema primitivo Heathkit/Zenith, tienen ROM desde la posición 0000H hasta la 42FFH. Digital Research, respondiendo a la presión del mercado, produjo una versión del CP/M que suponía que la RAM comenzaba en el lugar 4300H. Si se posee uno de estos sistemas, debe añadirse 4300H a todas las direcciones en los párrafos siguientes, *excepto* en lo que se refiere a las direcciones al principio de la memoria. Esto no será afectado por la presencia de la ROM en la memoria inferior.

Los valores individuales utilizados en los campos de la página base se describen en las secciones siguientes.

Arranque caliente El campo de tres bytes de *arranque caliente* contiene una instrucción para saltar a la parte superior de la RAM. Esta instrucción JMP transfiere el control al BIOS y dispara una operación de arranque caliente. Como se ha indicado anteriormente, un arranque caliente provoca que el CP/M vuelva a cargar el CCP y reconstruya el vector de situación para el disco actual por defecto. Si se necesita ocasionar un arranque caliente desde un programa cualquiera en lenguaje ensamblador, se codifica

```
JMP  0          ;Arranque caliente
```

BIOSPAGE El BIOS tiene varios puntos de entrada diferentes; sin embargo, todos ellos están ligados entre sí al principio del BIOS. Las primeras instrucciones del BIOS se parecen a las que siguen:

```
JMP  ENTRY1
JMP  ENTRY2
JMP  ENTRY3     ;y siguientes
```

Debido a la forma en que está montado el CP/M, la primera instrucción de salto comienza *siempre* en la frontera de una página. Recuérdese que una página equivale a 256 (100H) bytes de memoria, por lo que un límite de página es una dirección en la que los ocho bits menos significativos son cero. Por ejemplo, el vector de salto del BIOS (como se denomina al conjunto de JMP) puede empezar en una dirección tal como F200H o E600H. La dirección exacta está determinada por el tamaño del BIOS.

Mirando el BIOSPAGE puede determinarse el byte de dirección de la instrucción de arranque caliente del JMP más significativo y la dirección de página del vector de salto del BIOS.

IOBYTE El CP/M se basa en una filosofía de separar el mundo *material* del punto de vista *lógico* del mundo característico del CP/M. Esta filosofía se aplica también a los dispositivos orientados al procesamiento de caracteres que soporta CP/M.

El IOBYTE consiste en cuatro campos de dos bits que pueden ser utilizados para asignar un dispositivo material a cada uno de los lógicos. Es

importante entender que el IOBYTE en sí mismo es únicamente una estructura de datos pasiva. Las asignaciones prácticas suceden sólo cuando el controlador del dispositivo físico examina el IOBYTE, interpretando su contenido y seleccionando el controlador correcto para la cooperación del BIOS. Estos controladores de dispositivos están en código de bajo nivel (es decir, ligado íntimamente al lenguaje de la máquina) en el BIOS, quien prácticamente contacta y controla el dispositivo físico.

Los cuatro dispositivos *lógicos* que conoce el CP/M son:

1. *La consola.* Es el mecanismo a través del cual se comunica con el CP/M. Normalmente es un terminal con una pantalla y un teclado. La consola es un dispositivo bidireccional. Puede utilizarse como fuente de información (entrada) y como destino al cual se puede enviar información (salida).

   En terminología del CP/M, la consola se conoce con el nombre simbólico de "CON:". Obsérvese el ":" —éste diferencia el nombre del mecanismo del de un fichero en disco que podría llamarse "CON".

2. *El mecanismo de listado.* Normalmente es una impresora de algún tipo y se utiliza para hacer listados en papel. El CP/M ve a la impresora como un dispositivo de salida únicamente, lo que crea problemas en las impresoras que necesitan comunicar al CP/M que están ocupadas, pero este problema puede remediarse añadiendo instrucciones al controlador de la impresora.

   El nombre CP/M para este mecanismo lógico es de "LST:".

3. *La lectora de cinta de papel.* No es corriente encontrar una lectora de cinta de papel en uso hoy en día. Originalmente, el CP/M funcionaba con un sistema de desarrollo de microordenadores Intel, denominado MDS-800, y este sistema poseía una lectora de cinta de papel. Este mecanismo puede utilizarse sólo como fuente de información.

   El CP/M denomina a este mecanismo lógico "RDR:".

4. *La perforadora de cinta de papel.* Es también una reliquia de los primeros tiempos del CP/M y del MDS-800. En este caso, la perforada puede utilizarse sólo para salidas.

   El nombre del mecanismo lógico utilizado por el CP/M es "PUN:".

La distribución física de los campos del IOBYTE se muestra en la figura 4-2.

Cada campo de dos bits puede tomar uno de cuatro valores: 00, 01, 10 y 11. El valor particular puede ser interpretado por el BIOS como especificación de un dispositivo físico, según se indica en la tabla 4-1.

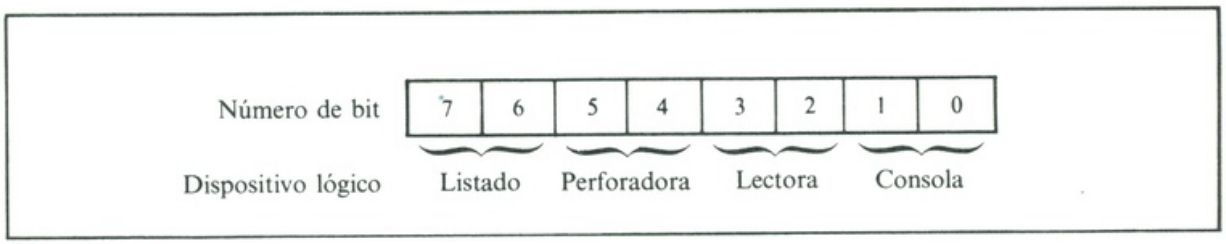


Figura 4-2. Disposición del IOBYTE.

A pesar de que la interpretación práctica del IOBYTE es llevada a cabo por el BIOS, el programa de utilidad STAT puede activar el IOBYTE utilizando los nombres de los dispositivos lógicos y físicos, y el PIP (programa de intercambio periférico) puede utilizarse para copiar datos de un dispositivo a otro. Además, se puede escribir un programa que cambie sencillamente el contenido del IOBYTE. Pero hay que tener cuidado, puesto que los cambios en el IOBYTE son efectivos inmediatamente.

Los valores del IOBYTE tienen los significados siguientes:

Consola (CON:)

00 Controlador del teletipo (TTY:)
Este controlador está pensado para conectarlo a un mecanismo de copia dura que se usa como consola principal.

01 Controlador de CRT (CRT:)
El controlador está pensado para ser conectado a un terminal CRT.

10 Modo de proceso por lotes (Batch)
Este es un caso especial. Se supone que los controladores adecuados serán llamados de forma que la entrada de la consola venga desde la lectora lógica (RDR:) y que la salida de la consola se envía al mecanismo de listado lógico (LST:).

Tabla 4-1. Valores del IOBYTE.

Dispositivo lógico	Dispositivo físico			
	00	01	10	11
Consola (CON:)	TTY:	CRT:	BAT:	UC1:
Lectora (RDR:)	TTY:	PTR:	UR1:	UR2:
Perforadora (PUN:)	TTY:	PTP:	UP1:	UP2:
Listado (LST:)	TTY:	CRT:	LPT:	UL1:

11 Consola definida para usuarios (UCI:)
El significado depende de la implementación del BIOS individual. Si, por ejemplo, se tiene una pantalla gráfica de alta resolución, puede arreglarse para ello colocando el IOBYTE para dirigir la salida de la consola a él. Puede hacerse que la entrada de la consola entre desde alguna tableta gráfica, palanca de mando, u otro dispositivo.

Lectora (RDR:)

00 Controlador del teletipo (TTY:)
Se refiere al dispositivo de lectura de cinta de papel que se encontraba frecuentemente en las consolas de teletipos.

01 Lectora de cinta de papel (PTR:)
Esto presupone algún tipo de dispositivo de entrada de gran velocidad conectado al sistema. Los sistemas modernos presentan raramente tal mecanismo, por lo que este montaje se utiliza a menudo para conectar la lectora lógica a la entrada de una línea de comunicación.

10 Lectora definida por el usuario #1 (UR1:)

11 Lectora definida por el usuario #2 (UR2:)
Estos dos montajes pueden utilizarse para dirigir el controlador físico a algún otro dispositivo especializado. Estos valores están incluidos únicamente porque en caso contrario no habrían sido asignados. Se utilizan raramente.

Perforadora (PUN:)

00 Controlador de teletipo (TTY:)
Se refiere a la perforadora de cinta de papel que se encontraba a menudo en las consolas de los teletipos.

01 Perforadora de cinta de papel (PTP:)
Esto presupone que existe algún tipo de perforadora de cinta de papel de alta velocidad conectada al sistema. De nuevo, es un caso raro, de forma que este mensaje se usa a menudo para conectar la perforadora lógica a la salida de una línea de comunicación.

10 Perforadora definida por el usuario #1 (UP1:)

11 Perforadora definida por el usuario #2 (UP2:)
Estos dos montajes corresponden a dos lectoras definidas por el usuario, pero prácticamente no se usan nunca.

Listado (LST:)

00 Controlador del teletipo (TTY:)
La salida se imprimirá en un teletipo.

01 Controlador del CRT (CRT:)
La salida se dirigirá a la pantalla en un terminal CRT.

10 Controlador de impresora de líneas (LPT:)
La salida irá a un mecanismo de impresión de alta velocidad. Aunque el nombre de *impresora de líneas* implica un tipo específico de hardware, puede ser cualquier tipo de impresora.

11 Dispositivo de listado definido por el usuario (UL1:)
Cualquiera que escriba, el BIOS puede preparar que este montaje cause una salida del dispositivo lógico de listado que vaya a un dispositivo distinto de la impresora principal.

Repetimos: El IOBYTE no se usa prácticamente por el cuerpo principal del CP/M. Es únicamente una estructura de datos pasivos que pueden ser manipulados por STAT. Si el IOBYTE posee algún efecto, depende totalmente de la implementación del BIOS particular.

CURUSER El campo CURUSER son los cuatro bits más significativos de su byte. Contiene los números de usuario seleccionado, establecidos por la orden CCP USER, a partir de una llamada específica al BDOS, o por un programa que sitúa esta parte en el valor deseado. Esta última forma de cambiar los números de usuario puede causar problemas de compatibilidad con versiones futuras del CP/M, de forma que debe utilizarse sólo en condiciones controladas.

CURDISK El campo CURDISK son los cuatro bits menos significativos del byte que se comparte con el CURUSER. Contiene el valor 0 si el disco en curso es A:, 1 si es B:, etc.

El campo CURDISK puede fijarse desde el CCP, por una petición al BDOS, o por un programa que altere este campo. Las objeciones hechas al CURUSER respecto a las compatibilidades son también válidas aquí.

BDOSE Este campo de tres bytes contiene una instrucción para saltar al punto de entrada del BDOS. Cuando se desea que el BDOS haga algo, se puede transferir la petición al BDOS colocando los valores adecuados en los registros y haciendo una llamada (CALL) a esta instrucción JMP. Utilizando un CALL, la dirección de retorno será situada en la pila. El JMP siguiente al BDOS no producirá información adicional alguna en la pila, que opera sobre una base "último en entrar, primero en salir", de esta forma, cuando el sistema vuelve del BDOS, volverá directamente al programa.

TOPRAM Puesto que el BDOS, al igual que el BIOS, empieza en una frontera de página, el byte más significativo de la dirección de la entrada del BDOS indica en qué página empieza el BDOS. Hay que restar 1 del valor del TOPRAM para obtener el mayor número de página que se puede usar en un programa. Obsérvese que cuando se utiliza esta técnica, se supone que se escribirá completamente encima del CCP hasta que se llegue en la memoria justo bajo el BDOS.

FCB1 y FCB2 Cuando conviene, el CCP toma los dos primeros parámetros que aparecen como parámetros de una orden (véase la próxima sección), intenta analizarlos como si fueran nombres de fichero y coloca los resultados en el FCB1 y el FCB2. Los resultados, en este contexto, significan que la letra del disco lógico se ha convertido en su representación en el FCB y el nombre y tipo de fichero, convertidos a mayúsculas, están situados en el FCB en los bytes correctos. Además, cualquier uso de "*" en el nombre de fichero se expande a uno o más signos de interrogación. Por ejemplo, un nombre de fichero de "abc*.*" se convertirá en un nombre de "ABC?????" y tipo de "???".

Obsérvese que el FCB2 empieza sólo 16 bytes por encima del FCB1, aunque un FCB normal tiene por lo menos una longitud de 33 bytes (36 bytes si se desea utilizar acceso aleatorio). En muchos casos, los programas necesitan sólo un único nombre de fichero. Por consiguiente, puede procederse a utilizar el FCB1 inmediatamente, sin tener en cuenta que se escribirá encima del FCB2.

En el caso del ejemplo del programa COPYFILE en las páginas anteriores, se requieren los nombres de dos ficheros. Antes de que el FCB1 pueda utilizarse, los 16 bytes del FCB2 deben trasladarse al esquema de FCB que se declara en el cuerpo del propio COPYFILE.

COMTAIL Los parámetros de orden o cola de la orden son cualquier cosa de la línea de orden *distinta* del nombre de dicha orden. Por ejemplo, la línea de orden COPYFILE se muestra a continuación:

```
A>copyfile tofile.type fromfile.typ display
```

El CCP toma los parámetros de la orden (convertidos a mayúsculas) y los almacena en el área COMTAIL.

COMTAIL$COUNT Es un cálculo binario en un único byte del número de caracteres de los parámetros de orden. El cálculo *no* incluye el CARRIAGE RETURN ni un espacio entre el nombre de la orden y los parámetros. Por ejemplo, si se introduce la línea de orden

```
A>PRINT ABC*.*
```

el COMTAIL$COUNT será seis, que es el número de caracteres de la cadena "ABC*.*".

COMTAIL$CHARS Son los caracteres reales de los parámetros de orden. Este campo no está inicializado a blancos, de forma que es preciso utilizar el COMTAIL$COUNT para detectar el final.

DMA$BUFFER En la figura 4-1, el DMA$BUFFER está en la misma área de memoria que COMTAIL. Este es un truco para ahorrar espacio, que funciona debido a que la mayoría de los programas procesan el contenido de la parte de parámetros antes de realizar cualquier entrada o salida en el disco.

El DMA$BUFFER es una memoria intermedia de un sector (luego tiene una longitud de 128 bytes). El uso de las siglas DMA (acceso directo a memoria) se refiere al MDS-800 de Intel. Este sistema posee hardware que puede trasladar datos a y desde disquetes yendo directamente a la memoria, obviando completamente a la CPU. El término se emplea todavía aunque se tenga una computadora que no utilice el DMA para la entrada/salida de sus discos. Puede sustituirse la idea de "la dirección a y de la cual los datos son leídos/escritos" en lugar del concepto de DMA.

Se puede solicitar al CP/M que utilice una dirección DMA distinta a la DMA$BUFFER, pero siempre que el CCP esté en control, la dirección de DMA se volverá a colocar aquí.

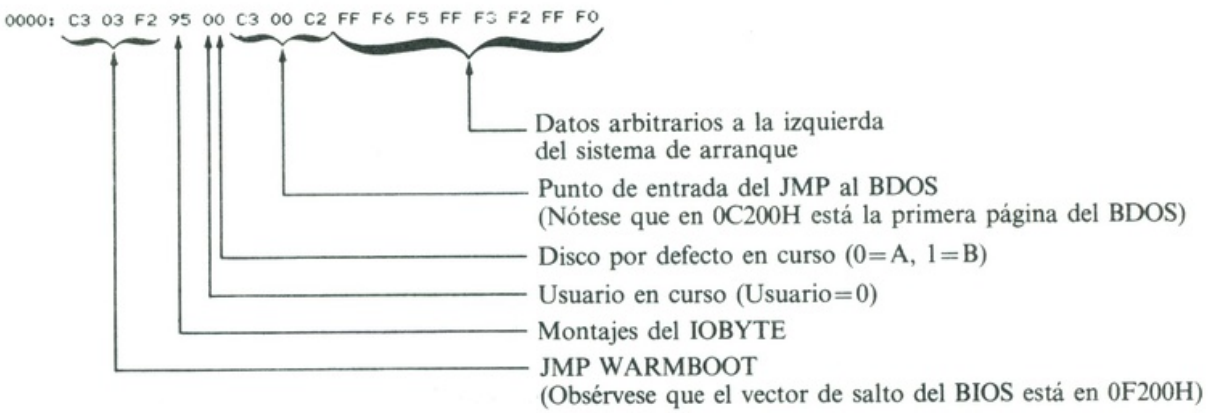
TPA Esta es el *área de programa transitoria* en la cual el CCP carga programas. El TPA se extiende hasta la base del BDOS.

El TPA es también la dirección de comienzo de la imagen de memoria que se guarda en el disco, siempre que se use la orden CCP SAVE.

Descarga de memoria de la página base

Lo que sigue son salidas de impresora que muestran los contenidos de la página base (los primeros 100H bytes de memoria) como los ve el programa COPYFILE.

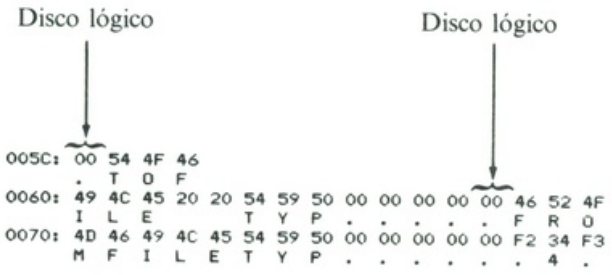
Un ejemplo de los primeros 16 bytes de memoria es el siguiente:



La línea de orden, como se recordará, era

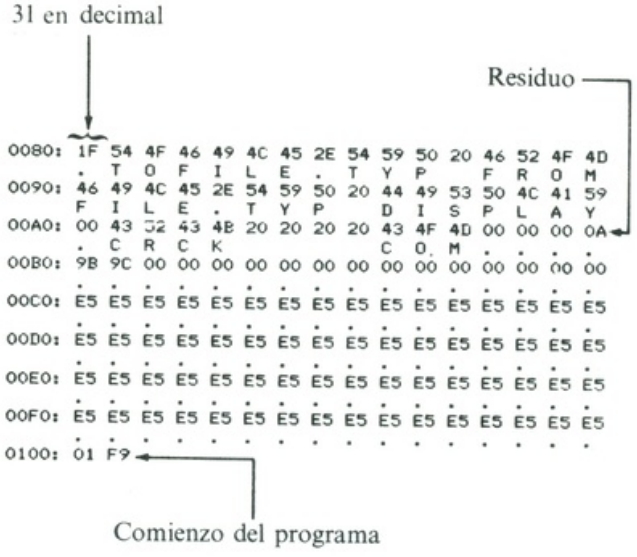
```
A>copyfile tofile.typ fromfile.typ display
```

Las áreas FCB1 y FCB2 serán determinadas por el CCP como sigue:



Como los discos lógicos no se especifican en los nombres de fichero de la línea de orden, el CCP ha situado el código del disco tanto en el FCB1 como en el FCB2 a 00H, con el significado "utilícese el disco por defecto". El nombre y tipo de fichero se han convertido en mayúsculas, separados e introducidos en el FCB en los sitios destinados a este fin.

La parte de parámetros de orden completa se ha almacenado en COMTAIL como sigue:



Como puede verse, la longitud de la parte parámetros es de 01FH (31 decimal). Esto va seguido inmediatamente por los propios caracteres parámetros-de-orden. Obsérvese que la parte de parámetros finaliza en el lugar 9FH. El resto de los datos que pueden verse son residuos de algunas operaciones anteriores realizadas por el CCP en el directorio. El nombre de

fichero CRCK.COM puede verse en una entrada al directorio, seguido de algunos 0E5H, que son espacios del directorio no utilizados.

Finalmente, en el lugar 0100H están los dos primeros bytes del programa.

Procesamiento de la parte de parámetros de orden (o cola de la orden)

Uno de los primeros problemas con los que hay que enfrentarse si se escribe un programa que pueda aceptar parámetros desde la parte de parámetros de una orden es procesarla, aislando cada uno de los parámetros. Debería utilizarse una subrutina estándar para este fin. Esta subrutina divide la línea de órdenes en parámetros individuales y devuelve un cálculo del número de parámetros, así como un apuntador a una tabla de direcciones. Cada dirección de esta tabla apunta por turno a una cadena terminada por un byte nulo. Cada parámetro se sitúa en una cadena distinta.

La figura 4-3 contiene el listado de esta subrutina, CTP (procesador de parámetros de orden).

```
0100                    ORG     100H
0100 CD3601     START:  CALL    CTP         ;Banco de pruebas para el CTP
0103 00                 NOP
                        ; Resto del programa

                ;       Esta subrutina fracciona la parte de parámetros de una orden,
                ;       situando cada uno de ellos en un área separada.
                ;
                ;       Parámetros de retorno:
                ;               A = 0 - No error (fija el indicador Z)
                ;               B = Cuenta del número de parámetros
                ;               HL -> Tabla de direcciones
                ;                       Cada dirección apunta a una cadena-parámetro
                ;                       terminada por un byte nulo.
                ;               Si se han especificado demasiados parámetros, A = TMP
                ;               Si uno de los parámetros es demasiado largo, A = PTL
                ;                       y D apunta al primer carácter de este parámetro
                ;                       en el área COMTAIL.
                ;
0080 =          COMTAIL         EQU     80H     ;Parte de parámetros de una orden en la página base
0080 =          COMTAIL$COUNT   EQU     COMTAIL ;Cuenta de car. existentes en la parte de parámetros de orden
0001 =          CTP$TMP         EQU     1       ;Código de error número excesivo de parámetros
0002 =          CTP$PTL         EQU     2       ;Código de error parámetro demasiado largo
                ;
                PTABLE:                         ;Tabla de apuntadores a parámetros
0104 0C01               DW      P1      ; Parámetro 1
0106 1A01               DW      P2      ; Parámetro 2
0108 2801               DW      P3      ; Parámetro 3
                        ; <--- Lugar para añadir unas direcciones de parámetros
010A 0000               DW      0       ; Terminal
                ;
                ;       Cadenas-parámetro.
                ;       El primer byte es 0, de forma que los parámetros
                ;       no utilizados son cadenas nulas.
                ;       El último byte de cada parámetro es 0 y se utiliza
                ;       para detectar parámetros demasiado largos.
```

Figura 4-3. Procesador de cola de órdenes (CTP).

```
010C 0001010101P1:      DB    0,1,1,1,1,1,1,1,1,1,1,1,1,1,0 ;Parámetro 1 y terminal
011A 0001010101P2:      DB    0,1,1,1,1,1,1,1,1,1,1,1,1,1,0 ;Parámetro 2 y terminal
0128 0001010101P3:      DB    0,1,1,1,1,1,1,1,1,1,1,1,1,1,0 ;Parámetro 3 y terminal
                        ; <--- Lugar para añadir más cadenas-parámetro
                ;
                CTP:                                        ;Punto de entrada principal <<<<<
0136 210401             LXI   H,PTABLE                      ;HL -> tabla de direcciones
0139 0E00               MVI   C,0                           ;Activación de la cuenta de parámetros
013B 3A8000             LDA   COMTAIL$COUNT                 ;Cuenta de caracteres
013E B7                 ORA   A                             ;Comprobación de existencia de parámetros
013F C8                 RZ                                  ;Salida (retorno de parámetros ya activados)
0140 E5                 PUSH  H                             ;Guardar en cabecera de pila para uso posterior
0141 47                 MOV   B,A                           ;B = cuenta de caracteres COMTAIL
0142 218100             LXI   H,COMTAIL+1                   ;HL -> caracteres en la parte de parámetros de orden.
                CTP$NEXTP:                                  ;Bucle parámetro siguiente
0145 E3                 XTHL                                ;HL -> tabla de direcciones
                                                            ;Cabecera de pila = apuntador COMTAIL
0146 5E                 MOV   E,M                           ;Obtención del byte LS de dirección de parámetro
0147 23                 INX   H                             ;Actualización del apuntador de direcciones
0148 56                 MOV   D,M                           ;Obtención del byte MS de dirección de parámetros
                                                            ;DE -> cadena-parámetro (o es 0)
0149 7A                 MOV   A,D                           ;Obtención de una copia del byte MS de dirección
014A B3                 ORA   E                             ;Combinación de los bytes MS y LS
014B CA8001             JZ    CTP$TMPX                      ;Demasiados parámetros-salida
014E 23                 INX   H                             ;Actualización del apuntador a la dirección siguiente
014F E3                 XTHL                                ;HL -> COMTAIL
                                                            ;Cabecera de pila--actualización de apuntador de dirección

                                ; En este punto se tiene
                                ; HL -> byte siguiente en la parte de parámetros de orden
                                ; DE -> primer byte de la siguiente cadena parámetro
                CTP$SKIPB:
0150 7E                 MOV   A,M                           ;Obtención el byte de parámetro siguiente
0151 23                 INX   H                             ;Actualización el apuntador a la parte de parámetros de orden
0152 05                 DCR   B                             ;Comprobación de existencia de nuevos caracteres
0153 FA7301             JM    CTPX                          ;No, salida
0156 FE20               CPI   ' '                           ;Comprobación existencia de blancos
0158 CA5001             JZ    CTP$SKIPB                     ;Si, saltar blancos
015B 0C                 INR   C                             ;Incrementación de la cuenta de parámetros
                CTP$NEXTC:
015C 12                 STAX  D                             ;Almacenamiento en cadena-parámetro
015D 13                 INX   D                             ;Actualización de apuntador a cadenas-parámetro
015E 1A                 LDAX  D                             ;Comprobación existencia byte siguiente
015F B7                 ORA   A                             ;Comprobación de si es un terminal
0160 CA7A01             JZ    CTP$PTLX                      ;Salida por parámetro demasiado largo
0163 AF                 XRA   A                             ;Añadir byte 00 al final del parámetro
0164 12                 STAX  D                             ;Almacenamiento en cadena-parámetro
0165 7E                 MOV   A,M                           ;Obtención de nuevo carácter de la parte de parámetros
0166 23                 INX   H                             ;Actualización de apuntador a zona de parámetros
0167 05                 DCR   B                             ;Comprobación existencia de caracteres
0168 FA7301             JM    CTPX                          ;No, salida
016B FE20               CPI   ' '                           ;Comprobación de parámetro terminal
016D CA4501             JZ    CTP$NEXTP                     ;Sí, ir a parámetro siguiente
0170 C35C01             JMP   CTP$NEXTC                     ;No, almacenamiento en cadena-parámetro
                ;
                CTPX:
                                                            ;Salida normal
0173 AF                 XRA   A                             A = 0 y activación de marca-Z
                ;
                CTPCX                                       ;Código de salida común
0174 E1                 POP   H                             ;Pop de la pila
0175 210401             LXI   H,PTABLE                      ;Retorno del apuntador a la tabla de dirección de parámetro
0178 B7                 ORA   A                             ;Asegurar la activación de la marca Z en la forma adecuada
0179 C9                 RET
                ;
                CTP$PTLX:                                   ;Salida por parámetro demasiado largo
017A 3E02               MVI   A,CTP$PTL                     ;Activación del código de error
017C EB                 XCHG                                ;DE -> parámetro erróneo
017D C37401             JMP   CTPCX                         ;Salida común
                ;
                CTP$TMPX:                                   ;Salida por exceso de parámetros
0180 3E01               MVI   A,CTP$TMP                     ;Activación del código de error
0182 C37401             JMP   CTPCX                         ;Salida común
                ;
0185                    END   START
```

Figura 4-3. (Continuación.)

Memoria disponible

Muchos programas necesitan utilizar toda la memoria disponible, y se precisa muy pronto en el programa fijar el apuntador de pila al extremo final de la RAM disponible. Como se ha indicado anteriormente, puede escribirse sobre el CCP, puesto que será recargado en el próximo arranque caliente.

La figura 4-4 muestra el código utilizado para fijar el apuntador de pila. Este programa determina la cantidad de memoria del TPA y fija el apuntador de pila en la parte superior de la RAM disponible.

Comunicación con el BIOS

Si se está escribiendo un programa de utilidad para su interacción con un BIOS adaptado, habrá ocasiones en que se necesite hacer una llamada *directa* al BIOS. Sin embargo, si el programa termina funcionando en su sistema operativo MP/M de Digital Research, se tendrán serios problemas si se intenta llamar directamente al BIOS. Entre otras cosas, se producirá una caída del sistema operativo.

Si se necesita realizar una llamada de este estilo y se es consciente de los peligros de utilizar llamadas directas al BIOS, la figura 4-5 muestra una forma de hacerlo.

Recuérdese que las primeras instrucciones del BIOS son el vector de saltos —una secuencia de instrucciones de JMP, una tras otra—. Antes de que se pueda hacer una llamada directa, es preciso saber *el desplazamiento relativo a la página* de la instrucción JMP particular a la que se desea ir. El vector de saltos del BIOS comienza siempre en una frontera de página, por lo que todo lo que se necesita es el byte menos significativo de su dirección.

Regreso al CP/M

Una vez ha corrido el programa, es preciso devolver el control otra vez al CP/M. Si el programa no ha escrito encima del CCP y ha dejado el apuntador de pila como estaba cuando se introdujo el programa, puede volverse directamente al CCP utilizando una instrucción RET.

La figura 4-6 muestra cómo un programa normal realizará todo esto si se usa una pila local, particular. La pila del CCP es demasiado pequeña; tiene sitio únicamente para 24 valores de 16 bits.

```
0007 =          TOPRAM  EQU     7       ;Byte más significativo del punto de entrada
                ;                               al BDOS
0000 3A0700             LDA     TOPRAM  ;Obtención del byte MS del punto de entrada al BDOS
0003 3D                 DCR     A       ;Retrasar una página
0004 2EFF               MVI     L,0FFH  ;Fjar el byte LS de la dirección final
0006 67                 MOV     H,A     ;HL = XXFFH
0007 F9                 SPHL            ;Fijar el apuntador de pila a partir de HL
```

Figura 4-4. Puesta del apuntador de pila a la cabeza de la RAM disponible.

```
                        ;       Utilícese esta técnica sólo en programas de utilidad CP/M.
                        ;       Los programas MP/M no lo permiten.
                        ;
0009 =                  CONIN   EQU     09H     ;Obtención de un carácter de entrada de la consola
                                                ;Está en la cuarta posición del vector de saltos
0002 =                  BIOSPAGE EQU    2       ;Dirección de página del BIOS
                        ;
                        ;       En este punto se realiza la llamada directa a CONIN
                        ;       CALL...
                        ;
0000 2E09                       MVI     L,CONIN ;Obtención del byte LS del punto de entrada CONIN
0002 CD0500                     CALL    BIOS    ;Ir a la subrutina de entrada al BIOS
                                ;... resto del programa...
                        ;
                        ;
                        BIOS:
0005 3A0200                     LDA     BIOSPAGE;Obtención de la página de vectores de salto del BIOS
0008 67                         MOV     H,A     ;HL -> punto de entrada
                                                ;(Se fija el byte LS antes de llegar aquí)
0009 E9                         PCHL            ;"JMP" al BIOS
                                                ;La dirección de retorno ya está
                                                ; en la pila
```

Figura 4-5. Realización de una llamada directa al BIOS.

La ventaja de volver directamente al CCP es la velocidad. Esto es cierto especialmente en un sistema de disco rígido, donde el tiempo que se necesita para realizar un arranque es bastante apreciable.

Si el programa ha escrito sobre el CCP, no se tiene otra opción que transferir el control al lugar 0000H y dejar que se realice un arranque caliente. Para ello, todo lo que se necesita es ejecutar

```
EXIT:   JMP   0         ;Arranque caliente
```

(Como indicación, si se está comprobando un programa y de repente se sale de nuevo al CP/M, lo probable es que inadvertidamente se haya enviado a la posición 0000H y se ha ejecutado un arranque caliente.)

```
                        ;       Nota: Este ejemplo supone que no
                        ;       se ha reescrito sobre el CCP
                        ;
0100                            ORG     100H    ;Comienzo en la TPA
                        START:
0100 210000                     LXI     H,0     ;Almacenamiento del apuntador de pila del CCP
0103 39                         DAD     SP      ; sumándolo a 0 en HL
0104 220F01                     SHLD    CCP$STACK
0107 314101                     LXI     SP,LOCAL$STACK
                        ;
                        ;       El cuerpo principal del programa está aquí
                        ;
                        ;       ... cuando se está preparado para volver
                        ;               al CCP...
010A 2A0F01                     LHLD    CCP$STACK       ;Obtención del apuntador de pila del CCP
010D F9                         SPHL                    ;Restauración del SP
010E C9                         RET                     ;Retorno al CCP
                        ;
010F                    CCP$STACK:      DS      2       ;Area de almacenaje del SP del CCP
0111                                    DS      48      ;Pila local
                        LOCAL$STACK:
0141                            END     START
```

Figura 4-6. Regreso al CCP al final del programa.

Lo que hace el BDOS
Llamadas a función del BDOS
Nomenclatura
Realización de una petición de función al BDOS

5

El sistema operativo básico de disco

El sistema operativo básico de disco es el corazón real del CP/M. Contrariamente al procesador de órdenes de consola, ha de estar en memoria todo el tiempo. Procura todos los servicios de entrada/salida a los programas del CP/M, incluido el CCP.

Como regla general, a menos que se esté escribiendo un programa de utilidad dependiente del sistema, debería usarse el BDOS para *todas* las entradas/salidas del programa. Si se salta al BDOS, probablemente se crearán problemas para el futuro.

Lo que hace el BDOS

El BDOS realiza todas las entradas/salidas del sistema. Estos servicios pueden agruparse en dos tipos de funciones:

Entrada/salida simple byte a byte
Es el envío y la recepción de datos entre la computadora y sus dispositivos lógicos —la consola, la "lectora" y la "perforadora" (o sus sustitutos), y la impresora.

Entrada/salida de ficheros de disco
Cubre tareas tales como la creación de nuevos ficheros, borrado de viejos, apertura de ficheros existentes y lectura y escritura de "registros" de 128 bytes de longitud en y desde estos ficheros.

El resto de este capítulo explica cada una de las funciones del BDOS, muestra cómo realizar cada petición al sistema operativo y proporciona información adicional para cada función. También se encuentran todos estos datos en el manual de Digital Research *Guía de interfases CP/M 2*, para descripciones estándar de estas funciones.

Llamadas a función del BDOS

Las llamadas a función del BDOS se describen en el orden de los números de código de la función. La figura 5-1 resume estas llamadas.

Nomenclatura

En la práctica, cuando se escriben programas que realizan llamadas al BDOS, es preciso incluir una serie de equivalencias para los números de código de la función del BDOS. En los siguientes ejemplos se hace referencia a estos valores, la forma en que se muestran en la figura 5-2 es tal como aparecen en los programas.

Los nombres de función utilizados para definir las equivalencias en la figura 5-2 son más cortos que los de la figura 5-1, con el fin de conseguir un equilibrio entre los nombres de función abreviados que se utilizan en la documentación de Digital Research y la necesidad de descripciones de funciones más claras.

Realización de una petición de función al BDOS

Todas las peticiones del BDOS se hacen enviando una instrucción CALL a la posición 0005H. También puede requerirse una función transfiriendo el control a la posición 0005H con la dirección de retorno en la pila.

Para decir al BDOS lo que se necesita que haga, hay que preparar los registros internos de la CPU para que contengan la información requerida antes de que se ejecute la instrucción CALL.

Código de la función	Descripción
	Entrada/salida simple byte a byte
0	Puesta a cero del BDOS y del sistema global
1	Lectura de un byte desde el teclado de la consola
2	Escritura de un byte en la pantalla de la consola
3	Lectura de un byte desde el dispositivo lógico de lectura
4	Escritura de un byte por el dispositivo de perforación
5	Escritura de un byte por el dispositivo lógico de impresión
6	Entrada/salida directa de la consola (no edición del estilo CCP)
7*	Lectura del contenido del IOBYTE
8*	Fijación de un nuevo valor del IOBYTE
9	Envío a la consola de una cadena terminada por $
10	Lectura de una cadena desde la consola en la memoria intermedia
11	Test de si una tecla de la consola espera para ser leída
12	Retorno del número de versión del CP/M
	Entrada/salida de ficheros de disco
13	Puesta a cero del sistema de disco
14	Selección del controlador del disco lógico especificado
15	Apertura de un fichero específico para lectura/escritura
16	Cierre de un fichero específico después de lectura/escritura
17	Búsqueda del directorio de ficheros para la primera coincidencia con el nombre de fichero
18	Búsqueda del directorio de ficheros para la siguiente coincidencia con el nombre de fichero
19	Borrado de fichero
20	Lectura del próximo registro secuencialmente
21	Escritura del próximo registro secuencialmente
22	Creación de un nuevo fichero con el nombre especificado
23	Cambio de nombre de un fichero
24	Indicación de los discos lógicos activos
25	Retorno del número de unidad de disco por defecto
26	Fijación de la dirección de DMA (dirección de lectura/escritura)
27	Retorno de la dirección de un vector de situación
28*	Fijar la unidad de disco lógico especificada al modo de sólo lectura
29	Indicación de los discos en curso que están en modo de sólo lectura
30	Fijar un fichero específico en modo sistema o de sólo lectura
31	Retorno de la dirección del bloque de parámetros del disco (DPR)
32*	Fijación/obtención del número de usuario en curso
33	Lectura de un registro aleatoriamente
34	Escritura de un registro aleatoriamente
35	Retorno del tamaño de fichero lógico (incluso para ficheros de acceso aleatorio)
36	Fijación del número de registro para la próxima lectura/escritura aleatoria
37	Puesta a cero de la unidad especificada
40	Escritura de un registro aleatorio lleno con ceros

* Estos no trabajan con el MP/M.

Figura 5-1. Llamadas a función del BDOS.

```
0000 =      B$SYSRESET      EQU     0       ;Puesta a cero del sistema
0001 =      B$CONIN         EQU     1       ;Lectura de un byte de la consola
0002 =      B$CONOUT        EQU     2       ;Escritura de un byte en la consola
0003 =      B$READIN        EQU     3       ;Lectura de byte de lectora
0004 =      B$PUNOUT        EQU     4       ;Escritura de byte por perforadora
0005 =      B$LISTOUT       EQU     5       ;Escritura de byte de impresora
0006 =      B$DIRCONIO      EQU     6       ;Entrada/salida directa por consola
0007 =      B$GETIO         EQU     7       ;Obtención del IOBYTE
0008 =      B$SETIO         EQU     8       ;Fijación del IOBYTE
0009 =      B$PRINTS        EQU     9       ;Impresión de una cadena por consola
000A =      B$READCONS      EQU     10      ;Lectura de una cadena por consola
000B =      B$CONST         EQU     11      ;Lectura del estado de la consola
000C =      B$GETVER        EQU     12      ;Obtención del número de versión del CP/M
000D =      B$DSKRESET      EQU     13      ;Puesta a cero del sistema de discos
000E =      B$SELDSK        EQU     14      ;Selección de disco
000F =      B$OPEN          EQU     15      ;Abrir fichero
0010 =      B$CLOSE         EQU     16      ;Cerrar fichero
0011 =      B$SEARCHF       EQU     17      ;Búsqueda de coincidencia con primer nombre
0012 =      B$SEARCHN       EQU     18      ;Búsqueda de coincidencia con nombre siguiente
0013 =      B$ERASE         EQU     19      ;Borrado de fichero
0014 =      B$READSEQ       EQU     20      ;Lectura secuencial
0015 =      B$WRITESEQ      EQU     21      ;Escritura secuencial
0016 =      B$CREATE        EQU     22      ;Creación de fichero
0017 =      B$RENAME        EQU     23      ;Cambio de nombre de fichero
0018 =      B$GETACTDSK     EQU     24      ;Obtención de los discos activos
0019 =      B$GETCURDSK     EQU     25      ;Obtención del disco por defecto en curso
001A =      B$SETDMA        EQU     26      ;Fijación de la dirección DMA (lectura/escritura)
001B =      B$GETALVEC      EQU     27      ;Obtención de la dirección del vector de situación
001C =      B$SETDSKRO      EQU     28      ;Puesta del disco en estado de sólo lectura
001D =      B$GETRODSKS     EQU     29      ;Obtención de los discos de sólo lectura
001E =      B$SETFAT        EQU     30      ;Fijación de atributos de fichero
001F =      B$GETDPB        EQU     31      ;Obtención de la dirección del bloque de parámetros
0020 =      B$SETGETUN      EQU     32      ;Activación/obtención del número de usuario
0021 =      B$READRAN       EQU     33      ;Lectura de acceso directo (aleatoria)
0022 =      B$WRITERAN      EQU     34      ;Escritura aleatoria
0023 =      B$GETFSIZ       EQU     35      ;Obtención de tamaño de ficheros
0024 =      B$SETRANREC     EQU     36      ;Fijación del número de registro de acceso directo (aleatorio)
0025 =      B$RESETD        EQU     37      ;Puesta a cero de la unidad
0028 =      B$WRITERANZ     EQU     40      ;Escritura aleatoria con rellenado-por-ceros
```

Figura 5-2. Equivalencias para números de código de funciones del BDOS.

En la llamada a una función específica, el número de código de la función que se requiere debe estar en el registro C.

Si se necesita llevar un valor de un solo byte al BDOS, tal como un carácter que se ha de enviar a la consola, entonces debe hacerse que este valor se encuentre en el registro E. Si el valor que se desea pasar al BDOS es un valor de 16 bits, tal como una dirección de memoria intermedia o del bloque de control de fichero (FCB), este valor deberá estar en el par de registros DE.

Cuando el BDOS devuelve un valor de un solo byte, tal como un carácter del teclado o un código de retorno que indique el éxito o fracaso de la función requerida, éste será enviado al registro A. Cuando el BDOS devuelve un valor de 16 bits, éste estará en el par de registros HL.

A la salida del BDOS, los registros A y L contendrán el mismo valor, igual que los registros B y H. Este convencionalismo extraño viene de los orígenes del CP/M en el PL/M (lenguaje programación/microprocesador), un lenguaje utilizado por Intel en sus sistemas MDS. Entonces el PL/M asentaba los fundamentos de lo que se conoce como convenios de llamada a registros.

El BDOS no da ninguna garantía acerca de los contenidos de los otros registros. Si se necesita preservar un valor que está en un registro, o bien se almacena dicho valor en la memoria o se introduce en la pila. El BDOS utiliza su propio espacio de pila, de forma que no hay necesidad de preocuparse por si agota nuestra pila.

Resumiendo, cuando se realiza una petición de función al BDOS se requiere un valor de un byte, el código y los parámetros de entrada y salida requeridos en la forma siguiente:

```
MVI     C,FUNCTION$CODE         ;C = código de función
MVI     E,SINGLE$BYTE           ;E = valor de un solo byte
CALL    BDOS                    ;Posición 5
                                ;A = retorno de código o valor
                                ;o HL = retorno de valor
```

Para aquellas peticiones de funciones que necesitan pasar una dirección al BDOS, la secuencia de llamada es:

```
MVI     C,FUNCTION$CODE         ;C = código de función
LXI     D,ADDRESS               ;DE = dirección
CALL    BDOS                    ;Posición 5
                                ;A = retorno de código o valor
                                ;o HL = retorno de valor
```

Si una petición de función incluye ficheros en disco, hay que indicar al BDOS la dirección del FCB que se ha creado para el fichero. (Véase capítulo 3 para descripciones del FCB.)

Muchas funciones de proceso de ficheros devuelven un valor al registro A que puede ser bien 0FFH, indicando que el fichero nombrado en el FCB no ha podido encontrarse, o el valor 0, 1, 2 ó 3. En el último caso, el BDOS está devolviendo lo que se llama un "código de directorio". El número es el de entrada del directorio que el BDOS contrastó con el nombre de fichero del FCB. En un momento dado, el BDOS tiene un sector de 128 bytes del directorio en la memoria. Cada entrada del directorio de ficheros es de 32 bytes, así que 4 de ellos (numerados 0, 1, 2 y 3) pueden ser procesados al mismo tiempo. El código de directorio indica cuál de ellos se ha emparejado con el del FCB.

Las referencias a los registros del CP/M en las descripciones siguientes aluden a sectores de 128 bytes. Es preciso no confundirlos con los registros lógicos utilizados en programas de aplicación. Los registros del CP/M han de considerarse siempre como sectores de 128 bytes.

Función 0: Puesta a 0 del sistema

Código de la función: C=00H
Parámetros de entrada: Ninguno
Parámetros de salida: No devuelve

Ejemplo

```
0000 =          B$SYSRESET      EQU     0       ;Puesta a cero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E00               MVI     C,B$SYSRESET    ;Fijación del código de función
0002 C30500             JMP     BDOS            ;Nota: Puede utilizarse un JMP ya que
                                                ; no se devolverá el control a este punto
```

Propósito La función de puesta a cero del sistema hace que el CP/M realice una puesta a cero completa, exactamente lo mismo que la función de arranque invocada cuando se transfiere el control al punto de WARMBOOT (véase figura 4-1).

Además de la puesta a cero del BDOS, esta función recarga el CCP, reconstruye los vectores de situación para los discos en curso, fija la dirección de DMA (utilizada por el CP/M para direccionar la memoria interna de lectura/escritura del disco) a 80H, marca todos los discos como si se encontraran en modo de lectura/escritura y transfiere el control al CCP. Este transfiere entonces su mensaje a la consola.

Notas Esta función es de la máxima utilidad cuando se trabaja en un lenguaje de alto nivel que no permite una instrucción de salto a una dirección absoluta de la memoria. Deberá utilizarse cuando el programa ha terminado y se necesita devolver el control al CP/M.

Función 1: Lectura de un byte de la consola

Código de función: C=01H
Parámetros de entrada: Ninguno
Parámetros de salida: A=Byte de datos de la consola

Ejemplo

```
0001 =          B$CONIN         EQU     1       ;Entrada de consola
0005 =          BDOS            EQU     5       ;Entrada al BDOS

0000 0E01               MVI     C,B$CONIN       ;Obtención código de función
0002 CD0500             CALL    BDOS
```

Propósito Esta función lee el próximo byte de datos del teclado de la consola y lo coloca en el registro A. Si el carácter de entrada es un carácter gráfico, será repetido en la consola. Los únicos caracteres de control que son repetidos son CARRIAGE RETURN, LINE FEED, BACKSPACE y TAB. En el caso de un carácter TAB, el BDOS envía tantos espacios como los que se necesitan para trasladar el cursor al próximo múltiplo de ocho columnas. Todos los caracteres de control restantes, incluido el CONTROL-C, entran pero no son repetidos.

Esta función comprueba también el CONTROL-S (XOFF) por si la salida de la consola ha de ser suspendida y el CONTROL-P (conmutador de impresión) por si la salida de la consola ha de ser enviada también al dispositivo de impresión. Si se encuentra el CONTROL-S, la salida a partir de este momento ha de ser suspendida hasta que se mecanografíe otro carácter. El CONTROL-P hará posible la repetición sobre impresora de la salida de la consola la primera vez que se pulsa y la hace imposible la segunda vez.

Si no existe ningún carácter de datos de entrada, esta función esperará hasta que haya una.

Notas

Esta función a menudo más que ayudar incordia, porque repite la entrada. Cuando se necesite entrada por consola a nivel de byte a byte, normalmente se desea suprimir esta repetición a la consola. Por ejemplo, es preciso saber que la "consola" es actualmente una línea de comunicaciones tal como un "modem" (?). Puede estarse intentando aceptar una palabra de paso que no debe ser repetida en la consola. O bien puede necesitarse leer un carácter de control del cursor que provoque un efecto lateral indeseable en el terminal si se repite allí.

Además, si se necesita más de un solo carácter de la consola, el programa será más fácil de utilizar si la persona que está en la consola puede beneficiarse totalmente de las ventajas de la edición de líneas estilo-CCP. Esto se puede hacer mejor utilizando la función de lectura de cadenas de la consola (código 10, 0AH).

La lectura de cadenas de caracteres de la consola es también muy útil para la entrada de caracteres únicos, especialmente cuando se está esperando una respuesta "Y" o "N" (sí o no). Si se usa la función de lectura de byte de la consola, el operador tendrá solamente una oportunidad para introducir los datos. Cuando se usa la lectura de cadenas, los usuarios tienen la posibilidad de mecanografiar un carácter, cambiar de opinión, retroceder un espacio y mecanografiar otro carácter.

Función 2: Escritura de un byte en la consola

Código de la función: C=02H
Parámetros de entrada: E=Byte de datos a lanzar a salida
Parámetros de salida: Ninguno

Ejemplo

```
0002 =          B$CONOUT        EQU     2       ;Escritura de byte en consola
0005 =          BDOS            EQU     5       ;Entrada al BDOS

0000 0E02               MVI     C,B$CONOUT      ;Código de función
0002 1E2A               MVI     E,'*'           ;E = byte de datos para salida
0004 CD0500             CALL    BDOS
```

Propósito Esta función saca el byte de datos del registro E a la consola. Igual que en la función 1, si el byte de datos es un carácter TAB, será expandido por el BDOS a la próxima columna que sea un múltiplo de ocho. El BDOS controla también la existencia de un carácter de entrada, y si es así, controla si es un CONTROL-S (caso en el cual la salida de la consola se suspende) o CONTROL-P (caso en el cual la repetición de la salida de la consola a la impresora se conecta o se desconecta).

Notas Pueden surgir problemas en el uso de esta función para enviar a la consola secuencias de control de dirección de cursor. Si se intenta sacar una dirección binaria real del cursor a la posición 9, el BDOS lo interpretará como un carácter TAB (ASCII código 9) y lo colocará debidamente con cero a ocho espacios. Si se necesita sacar valores binarios, se deberá fijar a uno el bit más significativo del carácter (úsese, por ejemplo, un ORI 80H) de forma que no sea tomado como el TAB ASCII.

Aquí hay dos subrutinas de propósito general, que pueden necesitarse para sacar mensajes. La primera, mostrada en la figura 5-3, saca desde una dirección especificada un mensaje terminado por un byte cero. El segundo, en la figura 5-4, realiza esencialmente lo mismo, *excepto* que la cadena del mensaje sigue inmediatamente después de la llamada a la subrutina.

Función 3: Lectura de byte de lectora

Código de función: C=03H
Parámetros de entrada: Ninguno
Parámetros de salida: A=Entrada de carácter

```
                        ;MSGOUT (salida de mensaje)
                        ;Salida de un mensaje terminado por un byte nulo.

                        ;Secuencia de llamadas
                        ;       MENSAJE:        DB      'Mensaje',0
                        ;               ;
                        ;       LXI     H,MESSAGE
                        ;       CALL    MSGOUT

                        ;Parámetros de salida
                        ;       HL -> byte nulo. Terminal

0002 =                  B$CONOUT        EQU     2       ;Escritura de un byte por consola
0005 =                  BDOS            EQU     5       ;Punto de entrada al BDOS

                        MSGOUT:
0000 7E                         MOV     A,M             ;Obtención del byte siguiente para salida
0001 B7                         ORA     A
0002 C8                         RZ                      ;Retorno cuando se da el byte nulo
0003 23                         INX     H               ;Actualización del apuntador-a-mensaje
0004 E5                         PUSH    H               ;Almacenamiento del apuntador actualizado
0005 5F                         MOV     E,A             ;Preparado para BDOS
0006 0E02                       MVI     C,B$CONOUT
0008 CD0500                     CALL    BDOS
000B E1                         POP     H               ;Recuperación del apuntador-a-mensaje
000C C30000                     JMP     MSGOUT          ;Vuelta atrás para carácter siguiente
```

Figura 5-3. Ejemplo de escritura de byte en la consola, salida de un mensaje terminado en byte cero de una dirección especificada.

```
                ;MSGOUTI (salida de mensaje en línea)
                ;Salida del mensaje terminado por un byte nulo que sigue
                ; a la llamada a MSGOUTI.

                ;Secuencia de llamadas
                ;       CALL    MSGOUTI
                ;       DB      'mensaje',0
                ;       ... instrucción siguiente

                ; Parámetros de salida
                ;       HL -> mensaje que sigue a la instrucción

0002 =          B$CONOUT        EQU     2       ;Escritura de un byte en consola
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                MSGOUTI:
0000 E1                 POP     H               ;HL -> mensaje
0001 7E                 MOV     A,M             ;Obtención byte de datos siguiente
0002 23                 INX     H               ;Actualización apuntador a mensaje
0003 B7                 ORA     A               ;Comprobación de byte nulo
0004 C20800             JNZ     MSGOUTIC        ;No, continúe
0007 E9                 PCHL                    ;Si, retorno a la instrucción siguiente
                                                ; después del mensaje en-línea

              MSGOUTIC:
0008 E5                 PUSH    H               ;Almacenamiento del apuntador-a-mensaje
0009 5F                 MOV     E,A             ;Preparado para BDOS
000A 0E02               MVI     C,B$CONOUT      ;Código de función
000C CD0500             CALL    BDOS
000F C30000             JMP     MSGOUTI         ;Vuelta atrás para carácter siguiente
```

Figura 5-4. Ejemplo de escritura de byte en la consola, salida de un mensaje terminado por un byte cero siguiendo la llamada a la subrutina.

Ejemplo

```
0003 =          B$READIN        EQU     3       ;Byte lector de la lectora
0005 =          BDOS            EQU     5       ;Entrada al BDOS

0000 0E03               MVI     C,B$READIN      ;Código de función
0002 CD0500             CALL    BDOS            ;A = byte lector
```

Propósito Esta función lee el próximo carácter del dispositivo "lector" lógico y lo sitúa en el registro A. En la práctica, el dispositivo físico al que se accede depende completamente de la configuración del BIOS. En algunos sistemas, no existe lectora: esta función devolverá un valor arbitrario, tal como 1AH (el carácter ASCII CONTROL-Z, utilizado por el CP/M para representar "Final de fichero").

El control no vuelve al programa de llamada hasta que el carácter ha sido leído.

Notas Como el dispositivo físico (si existe) utilizado cuando se envía esta petición depende totalmente de cada BIOS particular, no existe una posibilidad estándar por defecto para todas las implementaciones del CP/M. Esta es una de las partes más débiles del BDOS.

Habría que "conectar" el dispositivo de lectura por medio del software del BIOS a un puerto serie que puede ser utilizado para comunicación con otro sistema. Esto es únicamente una solución parcial al problema, porque

esta llamada a función no devuelve el control al programa hasta que se ha recibido un carácter de entrada. Existe un camino indirecto con el que se puede "escrutar" el dispositivo lector para ver si se ha recibido un carácter de entrada. Una vez realizada esta llamada a función, se pierde el control hasta que se ha recibido el próximo carácter; no hay ninguna función que corresponda a la lectura del estado de la consola (código de función 11, 0BH) que simplemente lea el estado y vuelva al programa.

Una posible solución es construir un temporizador en el controlador de lectura del BIOS que devuelva el control al programa con un valor imaginario en A si transcurre un período específico de tiempo sin ningún carácter de entrada. Pero esto lleva el problema de la elección del valor imaginario que se ha de utilizar. Si se considera enviar y recibir siempre ficheros que contengan pura información binaria, en ASCII no existe ningún carácter que no se pueda encontrar en un contexto legítimo. Por consiguiente, cualquier carácter imaginario que se pueda elegir podría ser también un dato real.

La solución más astuta es preparar un montaje del IOBYTE (que controle un mapa de dispositivo lógico a dispositivo físico) para conectar la consola a la línea de comunicación serie. Hecho esto, se puede hacer uso de la función lectura del estado de la consola, que no devolverá el estado de la consola física, sino el estado de la línea serie. El programa podrá entonces actuar adecuadamente si no se recibe ningún carácter en un período de tiempo establecido. La figura 5-11 muestra una subrutina que utiliza esta técnica en la función que fija el IOBYTE (código 8, 08H).

La figura 5-5 muestra un ejemplo de subrutina para leer líneas de datos desde el dispositivo lector. Lee caracteres desde la lectora, acumulándolos en la memoria hasta que se recibe o bien un LINE FEED o un número específico de caracteres. Obsérvese que el CARRIAGE RETURN se ignora, y que la línea de entrada está terminada por un byte 00H. El acuerdo de terminar las cadenas con bytes 00H y no con CARRIAGE RETURN se utiliza porque hace mucho más fácil la lógica del programa. Se adapta también a los convenios en lenguaje C.

Función 4: Escritura de byte por perforadora

Código de función: C=04H
Parámetros de entrada: E=Byte para salida
Parámetros de salida: Ninguno

Ejemplo

```
0004 =          B$PUNOUT        EQU     4       ;Escritura de byte por perforadora
0005 =          BDOS            EQU     5

0000 0E04               MVI     C,B$PUNOUT      ;Código de función
0002 1E2A               MVI     E,'*'           ;Byte de datos para salida
0004 CD0500             CALL    BDOS
```

```
                ;RL$RDR
                ;Lectura de línea del dispositivo de lectura.
                ;Se ignoran los retornos de carro y la lectura cuando
                ;se ha leído un número determinado de caracteres
                ;o bien cuando entra un salto de línea (LINE FEED).

                ;Nota: Una posible desventaja de esta subrutina es la no
                ;existencia de límite de tiempo. Esperará infinitamente
                ;si no llegan más caracteres al dispositivo lector.

                ;Secuencia de llamadas
                ;       LXI     H,BUFFER
                ;       LXI     B,MAXCOUNT
                ;       CALL    RL$RDR

                ;Parámetros de salida
                ;       HL -> cadena terminada en 00H
                ;       BC = cuenta atrás (0 si se ha leído el número máximo de caracteres)
                ;       E = último carácter leído

0003 =          B$READIN        EQU     3       ;Entrada de lectora
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

000D =          CR              EQU     0DH     ;Retorno de carro
000A =          LF              EQU     0AH     ;Salto de línea (terminal)

                RL$RDR:
0000 79                 MOV     A,C             ;Comprobación de cuenta a 0
0001 B0                 ORA     B               ;Si cuenta = 0 simular último
0002 5F                 MOV     E,A             ; carácter leído 00H
0003 CA2000             JZ      RL$RDRX         ;Sí, salida
0006 C5                 PUSH    B               ;Almacenar cuenta número máximo de caracteres
0007 E5                 PUSH    H               ;Almacenar apuntador a memoria intermedia

                RL$RDRI:                        ;Bucle para ignorar
0008 0E03               MVI     C,B$READIN
000A CD0500             CALL    BDOS            ;A = carácter de entrada
000D 5F                 MOV     E,A             ;Preservar una copia de los caracteres
000E FE0D               CPI     CR              ;Comprobación de si es un retorno de carro
0010 CA0800             JZ      RL$RDRI         ;Sí, ignorarse
0013 E1                 POP     H               ;Recuperación del apuntador a la memoria intermedia
0014 C1                 POP     B               ;Recuperación de cuenta de máximo
0015 FE0A               CPI     LF              ;Comprobación por si es un salto de línea
0017 CA2000             JZ      RL$RDRX         ;Sí, salida
001A 77                 MOV     M,A             ;No, almacenar carácter en la memoria intermedia
001B 23                 INX     H               ;Actualización del apuntador a la memoria intermedia
001C 0B                 DCX     B               ;Decrementación de contador
001D C30000             JMP     RL$RDR          ;Vuelta atrás para carácter siguiente

                RL$RDRX:
0020 3600               MVI     M,0             ;Memoria intermedia terminada por un byte nulo
0022 C9                 RET
```

Figura 5-5. Lectura de línea desde el dispositivo de lectura.

Propósito Esta función es un duplicado del byte "lector" de la lectora descrito anteriormente. Saca el carácter específico del registro E hasta el dispositivo lógico de perforación. De nuevo, el mecanismo físico utilizado, si existe, está determinado por el BIOS. No existe ningún conjunto estándar para este dispositivo; en algunos sistemas el dispositivo de perforación es un "cubo de bits", llamado así porque absorbe todos los datos que se le envían.

Notas Los problemas y posibles soluciones tratados para la llamada de la función del byte "lector" de la lectora pueden también aplicarse aquí. Una

diferencia, desde luego, es que esta función saca datos, por lo que el problema de un bucle indefinido esperando el próximo carácter es menos probable que ocurra. Sin embargo, si el dispositivo de perforación está conectado a una línea de comunicaciones, y si el hardware de salida no está preparado, el controlador de línea del BIOS esperará eternamente. Por desgracia, no existe ningún camino legítimo para luchar con este problema, ya que el BDOS no tiene ninguna llamada a función que controle si un mecanismo lógico está dispuesto para salida.

La figura 5-6 muestra una subrutina útil que saca una cadena terminada por el byte 00H a la perforadora. Cuando encuentra un LINE FEED, inserta un CARRIAGE RETURN en los datos de salida.

Función 5: Escritura de byte por dispositivo de listado

Código de función: C=05H
Parámetros de entrada: E=Bytes para salir
Parámetros de salida: Ninguno

```
                        ;WL$PUN
                        ;Escritura de una linea en el dispositivo de perforación
                        ;La salida termina cuando se encuentra un byte 00H.
                        ;Se lanza un retorno de carro cuando se encuentra
                        ;un salto de línea.

                        ;Secuencia de llamadas
                        ;       LXI     H,BUFFER
                        ;       CALL    WL$PUN

                        ;Parámetros de salida
                        ;       HL -> byte terminal 00H

0004 =          B$PUNOUT        EQU     4
0005 =          BDOS            EQU     5

000D =          CR              EQU     0DH     ;Retorno de carro
000A =          LF              EQU     0AH     ;Salto de línea

                WL$PUN:
0000 E5                 PUSH    H               ;Almacenamiento del apuntador a memoria intermedia
0001 7E                 MOV     A,M             ;Obtención del carácter siguiente
0002 B7                 ORA     A               ;Comprobación de si 00H
0003 CA2000             JZ      WL$PUNX         ;Si, salida
0006 FE0A               CPI     LF              ;Comprobación de salto de línea (00H)
0008 CC1600             CZ      WL$PUNLF        ;Si, O/P retorno de carro
000B 5F                 MOV     E,A             ;Carácter a enviar a la salida
000C 0E04               MVI     C,B$PUNOUT      ;Código de función
000E CD0500             CALL    BDOS            ;Salida de carácter
0011 E1                 POP     H               ;Recuperación del apuntador a memoria intermedia
0012 23                 INX     H               ;Incrementar para carácter siguiente
0013 C30000             JMP     WL$PUN          ;Salida carácter siguiente

                WL$PUNLF:                       ;Salto de línea encontrado
0016 0E04               MVI     C,B$PUNOUT      ;Código de función
0018 1E0D               MVI     E,CR            ;Lanzar un retorno de carro
001A CD0500             CALL    BDOS
001D 3E0A               MVI     A,LF            ;Creación de un salto de línea
001F C9                 RET                     ;Envío de LINE FEED

                WL$PUNX:                        ;Salida
0020 E1                 POP     H               ;Pop de la pila
0021 C9                 RET
```

Figura 5-6. Escritura de línea por el dispositivo de perforación.

Ejemplo

```
0005 =          B$LSTOUT        EQU     5       ;Escritura de byte por dispositivo de
0005 =          BDOS            EQU     5       ;listado

0000 0E05               MVI     C,B$LSTOUT      ;Código de función
0002 1E2A               MVI     E,'*'           ;Byte de datos para salida
0004 CD0500             CALL    BDOS
```

Propósito Esta función saca el byte especificado en el registro E por dispositivo lógico de listado. Como con la lectora y la perforadora, el dispositivo físico utilizado depende totalmente del BIOS.

Notas Uno de los mayores problemas asociados con esta función es que no se enfrenta con las condiciones de error de forma muy inteligente. No se puede estar seguro de qué mecanismo físico será utilizado como dispositivo de listado lógico, y la mayoría de las implementaciones estándar del BIOS causarán que el programa espere eternamente si la impresora no está dispuesta o se ha salido fuera del papel. El BDOS no está preparado para devolver algún tipo de estado de error para indicar que existe un problema con el mecanismo de listado. Por consiguiente, el BIOS tendrá que adaptarse para dominar esta situación.

La figura 5-7 es una subrutina que envía datos al mecanismo de listado. Como puede verse, es esencialmente una repetición de la figura 5-6, que configura la misma función para el dispositivo lógico de perforación.

Función 6: Entrada/salida directa de la consola

Código de función: C = 06H
Parámetros de entrada: E = 0FFH para entrada
E = Otros que no sean 0FFH para salida
Parámetros de salida: A = Byte de entrada o estado

Ejemplo

```
0006 =          B$DIRCONIO      EQU     6       ;Entrada/salida directa de la consola
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                                                ;Ejemplo de entrada por consola

0000 0E06               MVI     C,B$DIRCONIO    ;Código de función
0002 1EFF               MVI     E,0FFH          ;0FFH significa entrada
0004 CD0500             CALL    BDOS            ;A = 00 si no existe carácter a la
                                                ;espera
                                                ;A = NZ si existe

                                                ;Ejemplo de salida por consola

0007 0E06               MVI     C,B$DIRCONIO    ;Código de función
0009 1E2A               MVI     E,'*'           ;Distinto de 0FFH significa salida de
000B CD0500             CALL    BDOS            ;carácter
```

```
                ;WL$LST
                ;Escritura de una línea en el dispositivo de listado. La salida
                ;termina cuando se detecta un byte 00H.
                ;Se envía un retorno de carro cuando se detecta un
                ;salto de línea.

                ;Secuencia de llamadas
                ;       LXI     H,BUFFER
                ;       CALL    WL$LST

                ;Parámetros de salida
                ;       HL -> byte terminal 00H

0005 =          B$LSTOUT        EQU     5
0005 =          BDOS            EQU     5

000D =          CR              EQU     0DH     ;Retorno de carro
000A =          LF              EQU     0AH     ;Salto de línea

                WL$LST:
0000 E5                 PUSH    H               ;Almacenamiento del apuntador a memoria intermedia
0001 7E                 MOV     A,M             ;Obtención del carácter siguiente
0002 B7                 ORA     A               ;Comprobación de si 00H
0003 CA2000             JZ      WL$LSTX         ;Si, salida
0006 FE0A               CPI     LF              ;Comprobación de salto de línea (00H)
0008 CC1600             CZ      WL$LSTLF        ;Si, O/P retorno de carro
000B 5F                 MOV     E,A             ;Carácter a enviar a la salida
000C 0E05               MVI     C,B$LSTOUT      ;Código de función
000E CD0500             CALL    BDOS            ;Salida de carácter
0011 E1                 POP     H               ;Recuperación del apuntador a memoria intermedia
0012 23                 INX     H               ;Actualización para carácter siguiente
0013 C30000             JMP     WL$LST          ;Salida de carácter siguiente

                WL$LSTLF:                       ;Salto de línea encontrado
0016 0E05               MVI     C,B$LSTOUT      ;Código de función
0018 1E0D               MVI     E,CR            ;Enviar un retorno de carro a salida
001A CD0500             CALL    BDOS
001D 3E0A               MVI     A,LF            ;Creación de un salto de línea
001F C9                 RET                     ;Envío de LINE FEED

                WL$LSTX:                        ;Salida
0020 E1                 POP     H               ;Pop de la pila
0021 C9                 RET
```

Figura 5-7. Escritura de línea en el dispositivo de listado.

Propósito Esta función cumple una doble finalidad: introduce y saca caracteres de la consola. Sin embargo, no toma en consideración los caracteres de control normales ni las características particulares de la edición de líneas (tales como CONTROL-P y CONTROL-S) asociados normalmente con la entrada/salida de la consola, de aquí el nombre de "directo" (o "sin adornar" que describe Digital Research). Si el valor en el registro E *no* es 0FFH, entonces E contiene un carácter ASCII válido que sale a la consola. La lógica usada se entiende más fácilmente cuando se escribe en pseudocódigo:

```
si se trata de una petición de entrada (E = 0FFH)
     {
     si el estado de la consola indica que existe un carácter esperando
          {
          leer el carácter de la consola y retorno
          a la rutina de llamada con el carácter en A
          }
     sino (no hay ningún carácter esperando) vuelta
          a la rutina de llamada con A = 00
     }
sino (petición de salida)
     {
     enviar a la consola el carácter en E y volver
     a la rutina de llamada
     }
```

Notas

Esta función trabaja bien mientras que no haya que enviar nunca un valor 0FFH o se espere recibir un valor 00H. Si se necesita enviar o recibir datos puramente binarios, no se puede utilizar esta función, ya que estos valores pueden igualmente formar parte del flujo de datos.

Para entender por qué puede desearse enviar y recibir datos binarios, recuérdese que el "lector" lógico no tiene ningún método para que el operador compruebe su estado para ver si ha llegado un carácter de entrada. Todo lo que puede hacerse es intentar leer un carácter (byte lector de la lectora, código de función 3). Sin embargo, el BDOS no devolverá el control al operador hasta que llegue un carácter (lo que puede resultar muy largo). Una posibilidad es asignar lógicamente la consola a una línea de comunicación mediante el uso del IOBYTE (o cualquier otro medio similar) y entonces utilizar esta llamada de entrada/salida directa para enviar y recibir datos a y de la línea. Entonces se puede "registrar" la línea de comunicaciones y evitar el tener el programa esperando indefinidamente la llegada de un carácter de entrada. Un ejemplo de subrutina que utiliza esta técnica se muestra en la figura 5-11 sobre la fijación del IOBYTE (código de función 8).

La figura 5-8 muestra una subrutina que utiliza la entrada y la salida directas de la consola. Como este ejemplo es más complejo que cualquier otro de los mostrados hasta ahora, el programa utilizado para controlar la subrutina se ha incluido también.

```
                ;-------------------------------------------------------
                ; PROGRAMA DE PRUEBA
                ; Debido a la complejidad de esta subrutina, el verdadero
                ; programa base-de-comprobación se ha omitido en este ejemplo.
                ; Se supone que DDT o ZSID se utilizan para
                ; control de salida.
                ;-------------------------------------------------------

                        IF      1                       ;Cambiar a IF 0 para desactivar el programa
0100                    ORG     100H
0100 C31101             JMP     START                   ;Variables de paso activadas por DDT

0103 00         OPTIONS:        DB      0               ;Indicadores de opción
0104 41454900   TERMS:          DB      'A','E','I',0   ;Terminales
0108 05         BUFFER          DB      5               ;Número máximo de caracteres en memoria intermedia
0109 00                         DB      0               ;Contador
010A 6363636363                 DB      99,99,99,99,99  ;Bytes de datos
010F 6363                       DB      99,99

                START:
0111 210801             LXI     H,BUFFER        ;Obtención de la dirección de la memoria intermedia
0114 110401             LXI     D,TERMS         ;Dirección de la tabla de terminales
0117 3A0301             LDA     OPTIONS         ;Obtención de opciones activadas por DDT
011A 47                 MOV     B,A             ;Puesta en el registro correcto
011B CD2B01             CALL    RCS             ;Entrada de subrutina
011E CD3800             CALL    38H             ;Forzar punto de ruptura de DDT
0121 C31101             JMP     START           ;Nueva comprobación
                        ENDIF                   ;Final del programa de prueba

                ;RCS: Lectura de cadena de caracteres de la consola (utilizando entrada directa)
                ;Lee sobre una memoria intermedia una cadena de caracteres
                ; utilizando la entrada directa.
```

Figura 5-8. Lectura/escritura de cadenas de/a la consola utilizando entrada/salida directa.

```
                ; Opciones de soporte:
                ;       o Repetición (eco) de caracteres o no (si se repiten
                ;         debe enviarse un retorno de carro seguido de
                ;         un salto de línea)
                ;       o Arranque caliente cuando entra control-C o no
                ;       o Terminación de entrada por:
                ;               o Número máximo de caracteres de entrada
                ;               o o por derección de un carácter terminal

                ; Secuencia de llamada
                ;       LXI     H,BUFFER
                ;               La memoria intermedia tiene la estructura:
                ;               Mem. intermedia:  DB      10      Tamaño máximo
                ;                                 DB      0       Lectura real
                ;                                 DS      10+1    Area de memoria intermedia
                ;       MVI     B,OPTIONS         Opciones requeridas
                ;                                 (ver equivalencias)
                ;       LXI     D,TERMS           Apuntador al byte de final
                ;                                 de caracteres (00H),
                ;                                 cada uno de los cuales
                ;                                 es un terminal
                ;       CALL    RCS

                ; Parámetros de salida
                ;       Memoria intermedia: Actualizada con bytes de datos
                ;               y con el contador de caracteres de entrada.
                ;               (No se incluye el terminal.)
                ;       A = Código de terminación
                ;               0 =     Número máximo de caracteres de entrada.
                ;               NZ =    Terminal detectado.

0001 =          RCS$ECHO        EQU     0000$0001B      ;Caracteres de entrada a repetir
0002 =          RCS$ABORT       EQU     0000$0010B      ;Abortar si se da control-C
0004 =          RCS$FOLD        EQU     0000$0100B      ;Cambio de minúsculas a mayúsculas
0008 =          RCS$TERM        EQU     0000$1000B      ;DE -> fijación de carácter terminal

0006 =          B$DIRCONIO      EQU     6       ;I/O directa por consola
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0003 =          CTL$C           EQU     03H     ;Control-C
000D =          CR              EQU     0DH     ;Retorno de carro
000A =          LF              EQU     0AH     ;Salto de línea
0008 =          BS              EQU     08H     ;Retroceso

                RCS$ST:                         ;Tabla de terminales estándar
0124 0D                 DB      0DH             ;Retorno de carro
0125 0A                 DB      0AH             ;Final de línea
0126 00                 DB      0               ;Fin de tabla

                RCS$BSS:                        ;Secuencia de retrocesos destructiva
0127 08200800           DB      BS,' ',BS,0

                RCS:                            ;<<<<< Entrada principal
012B 23                 INX     H               ;HL -> contador real
012C 3600               MVI     M,0             ;Puesta al estado inicial
012E 2B                 DCX     H               ;HL -> máx. contador

                RCS$L:
012F E5                 PUSH    H               ;Almacenamiento del apuntador a memoria intermedia
0130 CD9201             CALL    RCS$GC          ;Obtención de carácter y ejecución de las opciones
                                                ; ECHO, ABORT y FOLD
                                                ;C = carácter de entrada
0133 E1                 POP     H               ;Recuperación del apuntador a memoria intermedia
0134 3E08               MVI     A,RCS$TERM      ;Comprobación de terminal especificado-por-el-usuario
0136 A0                 ANA     B               ;B = opciones
0137 C23D01             JNZ     RCS$UST         ;Terminales especificados-por-el-usuario
013A 112401             LXI     D,RCS$ST        ;Terminales estándar

                RCS$UST:
013D CDD401             CALL    RCS$CT          ;Comprobación existencia de terminal
0140 CA4C01             JZ      RCS$NOTT        ;No terminal
0143 47                 MOV     B,A             ;Conservación de los caracteres terminales

                RCS$MCI:                        ;(Número máximo de caracteres de entrada comparte este progama)
0144 0E00               MVI     C,0             ;Finalización de memoria intermedia
0146 CD7F01             CALL    RCS$SC          ;Almacenamiento de carácter
```

Figura 5-8. (Continuación.)

```
0149 78                 MOV     A,B             ;Recuperación de carácter terminal
014A B7                 'ORA    A               ;Activación de indicadores
014B C9                 RET

                RCS$NOTT:                       ;No terminal
014C 3E08               MVI     A,BS            ;Comprobación de retroceso
014E B9                 CMP     C
                                                ;Retroceso entrado
014F CA6001             JZ      RCS$BS          ;Almacenamiento de carácter en memoria intermedia
0152 CD7F01             CALL    RCS$SC          ;Actualización de contador
0155 CD8B01             CALL    RCS$UC          ;No máximo, obtención de otro carácter
0158 C22F01             JNZ     RCS$L           ;Simulación de carácter terminal
015B 0600               MVI     B,0             ;A = 0 por entrada del no máximo de caracteres
015D C34401             JMP     RCS$MCI

                RCS$BS:                         ;Retroceso entrado
0160 E5                 PUSH    H               ;Almacenamiento del apuntador a memoria intermedia
0161 23                 INX     H               ;HL -> cuenta real
0162 35                 DCR     M               ;Disminuir una unidad
0163 FA7A01             JM      RCS$NBS         ;Comprobación cuenta negativa
0166 212701             LXI     H,RCS$BSS       ;HL -> secuencia de retrocesos
0169 3E01               MVI     A,RCS$ECHO      ;No, comprobación de repetición
016B A0                 ANA     B               ;BS ha sido repetida
016C CA7001             JZ      RCS$BSNE        ;No, la entrada BS no repetida
016F 23                 INX     H               ;Saltarse el retroceso inicial

                RCS$BSNE:
0170 C5                 PUSH    B               ;Almacenar opciones y carácter
0171 D5                 PUSH    D               ;Almacenar apuntador a tabla de terminales
0172 CDF601             CALL    WCS             ;Escritura de cadena por consola
0175 D1                 POP     D               ;Recuperación del apuntador a tabla de terminales
0176 C1                 POP     B               ;Recuperación de opciones y carácter
0177 C37B01             JMP     RCS$BSX         ;Salida desde retroceso lógico

                RCS$NBS:
017A 34                 INR     M               ;Puesta a 0 del contador

                RCS$BSX:
017B E1                 POP     H               ;Recuperación del apuntador a memoria intermedia
017C C32F01             JMP     RCS$L           ;Obtención del carácter siguiente

                RCS$SC:                         ;Guardar en la memoria intermedia el carácter en C
                                                ;HL -> apuntador a memoria intermedia
017F D5                 PUSH    D               ;Almacenar apuntador a tabla de terminales
0180 E5                 PUSH    H               ;Almacenar apuntador a memoria intermedia
0181 23                 INX     H               ;HL -> cuenta de un carácter en memoria intermedia
0182 5E                 MOV     E,M             ;Obtención de la cuenta real
0183 1C                 INR     E               ;Cuenta de puntos 0 hasta el primer byte de datos
0184 1600               MVI     D,0             ;Transferir el valor de contador a palabra
0186 19                 DAD     D               ;HL -> siguiente byte de datos libre
0187 71                 MOV     M,C             ;Almacenamiento de byte de datos
0188 E1                 POP     H               ;Recuperación del apuntador de memoria intermedia
0189 D1                 POP     D               ;Recuperación del apuntador a
                                                ; la tabla de terminales
018A C9                 RET

                RCS$UC:                         ;Actualización del contador de memoria y comprobación de máx.
                                                ;Retorno Z activo si = máx., NZ
                                                ; si no HL -> memoria intermedia en entrada
018B E5                 PUSH    H               ;Almacenar apuntador a memoria intermedia
018C 7E                 MOV     A,M             ;Obtención máximo de cuenta
018D 23                 INX     H               ;HL -> cuenta real
018E 34                 INR     M               ;Incremento cuenta real
018F BE                 CMP     M               ;Comparación máxima con real
0190 E1                 POP     H               ;Recuperación apuntador memoria intermedia
0191 C9                 RET                     ;Marca Z activada

                RCS$GC:                         ;Obtención de carácter y ejecución de las opciones
                                                ; ECHO, ABORT y FOLD
0192 D5                 PUSH    D               ;Almacenar apuntador a tabla terminales
0193 E5                 PUSH    H               ;Almacenar apuntador a memoria intermedia
0194 C5                 PUSH    B               ;Almacenar indicadores de opción
```

Figura 5-8. (Continuación.)

```
                RCS$WT:
0195 0E06               MVI     C,B$DIRCONIO    ;Código de función
0197 1EFF               MVI     E,OFFH          ;Especificar entrada
0199 CD0500             CALL    BDOS
019C B7                 ORA     A               ;Comprobación de datos esperando
019D CA9501             JZ      RCS$WT          ;Vuelta atrás y espera
01A0 C1                 POP     B               ;Recuperación de indicadores de opción
01A1 4F                 MOV     C,A             ;Almacenamiento de byte de datos
01A2 3E02               MVI     A,RCS$ABORT     ;Comprobación de opción abortar activada
01A4 A0                 ANA     B
01A5 CAAE01             JZ      RCS$NA          ;No aborto
01A8 3E03               MVI     A,CTL$C         ;Comprobación de control-C
01AA B9                 CMP     C
01AB CA0000             JZ      0               ;Arranque caliente

                RCS$NA:
01AE 3E04               MVI     A,RCS$FOLD      ;Comprobación de actuación sustitución de caracteres (FOLD)
01B0 A0                 ANA     B
01B1 C4E501             CNZ     TOUPPER         ;Convertir a mayúsculas
01B4 3E01               MVI     A,RCS$ECHO      ;Comprobación petición de repetición
01B6 A0                 ANA     B
01B7 CAD101             JZ      RCS$NE          ;No se requiere repetición
01BA C5                 PUSH    B               ;Almacenamiento de opciones y carácter
01BB 59                 MOV     E,C             ;Transferencia carácter para salida
01BC 0E06               MVI     C,B$DIRCONIO    ;Código de función
01BE CD0500             CALL    BDOS            ;Repetición de carácter
01C1 C1                 POP     B               ;Recuperación de opciones y carácter
01C2 3E0D               MVI     A,CR            ;Comprobación de retorno de carro
01C4 B9                 CMP     C
01C5 C2D101             JNZ     RCS$NE          ;No
01C8 C5                 PUSH    B               ;Almacenar opciones y carácter
01C9 0E06               MVI     C,B$DIRCONIO    ;Código de función
01CB 1E0A               MVI     E,LF            ;Enviar a salida salto de línea
01CD CD0500             CALL    BDOS
01D0 C1                 POP     B               ;Recuperar opciones y carácter

                RCS$NE:
01D1 E1                 POP     H               ;Recuperar apuntador a memoria intermedia
01D2 D1                 POP     D               ;Recuperar tabla de terminales
01D3 C9                 RET                     ;Carácter en C

                RCS$CT:                         ;Comprobación de terminal
                                                ;C = carácter entrado
                                                ;DE -> cadena de caracteres
                                                ; terminales (byte 00H)
                                                ;Devuelve estado Z si no
                                                ; encuentra coincidencia, NZ si la
                                                ; encuentra (con A = C = último
                                                ; carácter)
01D4 D5                 PUSH    D               ;Almacenar apuntador de tabla

                RCS$CTL:                        ;Obtención del siguiente carácter terminal
01D5 1A                 LDAX    D               ;Comprobación final de tabla
01D6 B7                 ORA     A               ;No coincidencia con ningún terminal
01D7 CAE201             JZ      RCS$CTX         ;Comparación con carácter de entrada
01DA B9                 CMP     C               ;Coincidencia con terminal
01DB CAE201             JZ      RCS$CTX         ;Movimiento hacia el terminal siguiente
01DE 13                 INX     D               ;Bucle para probar el siguiente
01DF C3D501             JMP     RCS$CTL         ; carácter de la tabla

                RCS$CTX:
01E2 B7                 ORA     A               ;Comprobación de salida por terminal
                                                ;En este punto A será 0 si se
                                                ; ha alcanzado el final
                                                ; de la tabla, 0 NZ si se ha
                                                ; encontrado una coincidencia.
                                                ; El indicador Z será
01E3 D1                 POP     D               ; activado
01E4 C9                 RET                     ;Recuperación del apuntador de tabla

                ; TOUPPER - Cambio de minúsculas por mayúsculas
                ;         C = Carácter de entrada y salida
```

Figura 5-8. (Continuación.)

```
                TOUPPER:
01E5 3E60               MVI     A,'a'-1         ;Comprobación de necesidad del cambio
01E7 B9                 CMP     C               ;Comprobación con el carácter de entrada
01E8 D2F501             JNC     TOUPX           ;No, carácter es < = "a"-1
01EB 3E7A               MVI     A,'z'           ;Puede ser, carácter = > "a"
01ED B9                 CMP     C
01EE DAF501             JC      TOUPX           ;No, carácter > "z"
01F1 3EDF               MVI     A,0DFH          ;Cambio de carácter
01F3 A1                 ANA     C
01F4 4F                 MOV     C,A             ;Devolución del carácter cambiado

                TOUPX:
01F5 C9                 RET

                ;WCS - Escritura de cadena en consola (utilizando E/S directa)
                ;La salida termina cuando se encuentra un byte 00H.
                ;Se envía a la salida un retorno de carro cuando se detecta
                ;un salto de línea.

                ;Secuencia de llamada
                ;       LXI     H,BUFFER
                ;       CALL    WCS

                ;Parámetros de salida
                ;       HL -> byte terminal 00H

                WCS:
01F6 E5                 PUSH    H               ;Almacenamiento del apuntador a memoria intermedia
01F7 7E                 MOV     A,M             ;Obtención carácter siguiente
01F8 B7                 ORA     A               ;Comprobación de 00H
01F9 CA1602             JZ      WCSX            ;Si, salida
01FC FE0A               CPI     LF              ;Comprobación de salto de línea
01FE CC0C02             CZ      WCSLF           ;Sí, salida de un retorno de carro
0201 5F                 MOV     E,A             ;Carácter para enviar a salida
0202 0E06               MVI     C,B$DIRCONIO    ;Código de función
0204 CD0500             CALL    BDOS            ;Salida de carácter
0207 E1                 POP     H               ;Recuperación de apuntador de memoria intermedia
0208 23                 INX     H               ;Actualización para carácter siguiente
0209 C3F601             JMP     WCS             ;Salida del carácter siguiente

                WCSLF:                          ;Detectado un salto de línea
020C 0E06               MVI     C,B$DIRCONIO    ;Código de función
020E 1E0D               MVI     E,CR            ;Salida de un retorno de carro
0210 CD0500             CALL    BDOS
0213 3E0A               MVI     A,LF            ;Creación de salto de línea
0215 C9                 RET                     ;Salida de un LF

                WCSX:                           ;Salida
0216 E1                 POP     H               ;Reajuste de la pila
0217 C9                 RET
```

Figura 5-8. (Continuación.)

Función 7: Obtención del montaje IOBYTE

Código de función: C=07H
Parámetros de entrada: Ninguno
Parámetros de salida: A=Valor del IOBYTE

Ejemplo

```
0007 =          B$GETIO         EQU     7       ;Obtención del IOBYTE
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E07               MVI     C,B$GETIO       ;Código de función
0002 CD0500             CALL    BDOS            ;A = IOBYTE
```

Propósito Esta función coloca el valor en curso del IOBYTE en el registro A.

Notas Como vimos en el capítulo 4, el IOBYTE es un medio de asociar los dispositivos lógicos del CP/M (consola, lectora, perforadora y listado) con los dispositivos físicos soportados por un BIOS particular. El uso del IOBYTE es completamente opcional. El CP/M, según dice textualmente la *Guía de modificaciones de CP/M 2.0,* "tolera la existencia el IOBYTE en la posición 0003H".

En la práctica, la utilidad STAT que proporciona Digital Research posee algunos mecanismos que sitúan diferentes valores en el IOBYTE desde la consola del sistema.

La figura 5-9 resume la estructura del IOBYTE. Una descripción más detallada se dio en el capítulo 4.

Cada campo de dos bits puede tomar uno de cuatro valores: 00, 01, 10 y 11. El valor puede ser interpretado por el BIOS como un dispositivo físico específico, según se muestra en la tabla 4-1.

La figura 5-10 muestra las equivalencias que se utilizan con referencia al IOBYTE. Como puede verse, los valores que se muestran se han declarado utilizando el operador SHL (desplazamiento a la izquierda) del ensamblador de Digital Research. Esto no es más que un recordatorio de que los valores están estructurados de esta forma en el propio IOBYTE.

Función 8: Fijación del IOBYTE

Código de función: C=08H
Parámetros de entrada: E=Valor nuevo del IOBYTE
Parámetros de salida: Ninguno

Ejemplo El listado siguiente muestra cómo asignar el dispositivo lógico de lectura al controlador de consola del BIOS. Este utiliza algunas equivalencias de la figura 5-10.

```
0007 =          B$GETIO         EQU     7       ;Obtención del IOBYTE
0008 =          B$SETIO         EQU     8       ;Fijación del IOBYTE
0005 =          BDOS            EQU     5       ;Punto de entrada del BDOS

000C =          IO$RDRM         EQU     0000$1100B  ;Máscara de bits de la lectora
0008 =          IO$RUR1         EQU     2 SHL 2     ;Selección de usuario de lectora

                ;Este ejemplo muestra cómo asignar la lectora lógica
                ;a la lectora definida por el usuario #1 (UR1:)

0100                    ORG     100H
0100 0E07               MVI     C,B$GETIO       ;Primero, obtención IOBYTE en curso
0102 CD0500             CALL    BDOS
0105 E6F3               ANI     (NOT IO$RDRM) AND OFFH ;Conservación de todos los bits
                                                       ; excepto los de lectora
0107 F608               ORI     IO$RUR1         ;OR, en se fijan de nuevo
0109 5F                 MOV     E,A             ;Preparando para fijar IOBYTE
010A 0E08               MVI     C,B$SETIO       ;Fijar nuevo valor
010C CD0500             CALL    BDOS
```

Número de bit	7	6	5	4	3	2	1	0
Dispositivo lógico	Listado		Perforadora		Lectora		Consola	

Figura 5-9. Estructura del IOBYTE.

Propósito Esta función sitúa en el IOBYTE un nuevo valor que está dado en el registro E. A causa del campo de bits individual del IOBYTE, normalmente se utilizará la función de obtención del IOBYTE, se cambiarán algunos bits del valor en curso, y entonces se llamará la función de fijación del IOBYTE.

Notas Se pueden utilizar las funciones de fijación del IOBYTE, de obtención IOBYTE y de entrada/salida directa de consola juntas para crear un pequeño programa que transforme el sistema de cálculo en un terminal inteligente. Todos los datos que se mecanografíen en el teclado pueden enviarse a una línea de comunicación serie de otro ordenador, y todos los datos recibidos en la línea pueden ser enviados a la pantalla.

```
                ;Equivalencias IOBYTE
                ;Sirven para acceso al IOBYTE

                ;Máscara para aislar dispositivos específicos.
                ;Inversamente puede utilizarse para proteger todos los dispositivos
                ; excepto uno específico

0003 =          IO$CONM EQU     0000$0011B      ;Máscara de consola
000C =          IO$RDRM EQU     0000$1100B      ;Máscara de lectora
0030 =          IO$PUNM EQU     0011$0000B      ;Máscara de perforadora
00C0 =          IO$LSTM EQU     1100$0000B      ;Máscara de impresora

                                                ;Valores de consola
0000 =          IO$CTTY EQU     0               ;Consola -> TTY:
0001 =          IO$CCRT EQU     1               ;Consola -> CRT:
0002 =          IO$CBAT EQU     2               ;Consola (entrada) <- RDR:
                                                ;Consola (salida) -> LST:
0003 =          IO$CUC1 EQU     3               ;Consola -> UC1: (consola de usuario 1)

                                                ;Valores de lectora
0000 =          IO$RTTY EQU     0 SHL 2         ;Lectora <- TTY:
0004 =          IO$RRDR EQU     1 SHL 2         ;Lectora <- RDR:
0008 =          IO$RUR1 EQU     2 SHL 2         ;Lectora <- UR1: (lectora usuario 1)
000C =          IO$RUR2 EQU     3 SHL 2         ;Lectora <- UR2: (lectora usuario 2)

                                                ;Valores de perforadora
0000 =          IO$PTTY EQU     0 SHL 4         ;Perforadora -> TTY:
0010 =          IO$PPUN EQU     1 SHL 4         ;Perforadora -> PUN:
0020 =          IO$PUP1 EQU     2 SHL 4         ;Perforadora -> UP1: (perforadora usuario 1)
0030 =          IO$PUP2 EQU     3 SHL 4         ;Perforadora -> UP2: (perforadora usuario 2)

                                                ;Valores de listado
0000 =          IO$LTTY EQU     0 SHL 6         ;Listado -> TTY:
0040 =          IO$LCRT EQU     1 SHL 6         ;Listado -> CRT:
0080 =          IO$LLPT EQU     2 SHL 6         ;Listado -> LPT: (impresora de líneas físicas)
00C0 =          IO$LUL1 EQU     3 SHL 6         ;Listado -> UL1: (impresora usuario 1)
```

Figura 5-10. Equivalencias IOBYTE.

La figura 5-11 muestra este programa e ilustra el uso de todas estas funciones.

Para que este programa funcione correctamente, el BIOS deberá controlar el IOBYTE y detectar si la consola lógica está conectada a la consola física (con el IOBYTE situado en TTY:) o al lado de entrada de la línea de comunicaciones serie (con el IOBYTE puesto en RDR:).

La figura 5-11 muestra cómo utilizar las funciones de obtención y fijación del IOBYTE para realizar un simple emulador de terminal. Para que este ejemplo funcione, el BIOS deberá detectar el valor de la consola como 3 (IO$CUCI) y conectar las funciones de estado de la consola y entrada y salida de la misma a la línea de comunicaciones.

```
0006 =          B$DIRCONIO      EQU     6       ;Entrada/salida directa de consola
0007 =          B$GETIO         EQU     7       ;Obtención del IOBYTE
0008 =          B$SETIO         EQU     8       ;Fijación del IOBYTE
000B =          B$CONST         EQU     11      ;Obtención del estado de la consola
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0003 =          IO$CONM EQU     0000$0011B      ;Máscara de consola para IOBYTE
0001 =          IO$CCRT EQU     1               ;Consola -> CRT:
0003 =          IO$CUC1 EQU     3               ;Consola -> consola de usuario #1

                TERM:
0000 CD2A00             CALL    SETCRT          ;Conexión consola -> CRT:

                TERM$CKS:
0003 CD5200             CALL    CONST           ;Obtención estado CRT
0006 CA2400             JZ      TERM$NOKI       ;No consola de entrada
0009 CD4B00             CALL    CONIN           ;Obtención de carácter por teclado
000C CD3000             CALL    SETCOMM         ;Conexión consola -> línea com.
000F CD4500             CALL    CONOUT          ;Salida a la línea de com.

                TERM$CCS:                       ;Comprobación estado com.
0012 CD5200             CALL    CONST           ;Obtención del estado de la "consola"
0015 CA0000             JZ      TERM            ;Ningún carácter de entrada
0018 CD4B00             CALL    CONIN           ;Obtención de carácter de entrada por com.
001B CD2A00             CALL    SETCRT          ;Conexión consola -> CRT:
001E CD4500             CALL    CONOUT          ;Salida a CRT
0021 C30300             JMP     TERM$CKS        ;Bucle atrás para comprobar estado del teclado

                TERM$NOKI:
0024 CD3000             CALL    SETCOMM         ;Conexión consola -> línea com.
0027 C31200             JMP     TERM$CCS        ;Bucle atrás para comprobar estado com.

                SETCRT:                         ;Conexión consola -> CRT:
002A F5                 PUSH    PSW             ;Almacenar posible carácter dato
002B 0601               MVI     B,IO$CCRT       ;Conexión consola -> CRT:
002D C33300             JMP     SETCON          ;Código común

                SETCOMM:                        ;Conexión consola -> línea com.
0030 F5                 PUSH    PSW             ;Almacenar posible carácter dato
0031 0603               MVI     B,IO$CUC1       ;Conexión consola -> línea com.
                                                ;Salto a SETCON

                SETCON:                         ;Activación dispositivo de consola
                                                ;Nuevo código en B (en los bits 1,0)
0033 C5                 PUSH    B               ;Almacenar código
0034 0E07               MVI     C,B$GETIO       ;Obtención IOBYTE en curso
0036 CD0500             CALL    BDOS
0039 E6FC               ANI     (NOT IO$CONM) AND OFFH  ;Conservar todo menos la consola
003B C1                 POP     B               ;Recuperar el código requerido
003C B0                 ORA     B               ;OR de los nuevos bits
003D 5F                 MOV     E,A             ;Preparado para fijación de valor
003E 0E08               MVI     C,B$SETIO       ;Código de función
0040 CD0500             CALL    BDOS
0043 F1                 POP     PSW             ;Recuperar posible carácter dato
0044 C9                 RET
```

Figura 5-11. Emulador de terminal sencillo.

```
                CONOUT:
0045 5F                 MOV     E,A             ;Obtener byte de datos para salida
0046 0E06               MVI     C,B$DIRCONIO    ;Código de función
0048 C30500             JMP     BDOS            ;BDOS devuelve control a la rutina que ha llamado a CONOUT

                CONIN:
004B 0E06               MVI     C,B$DIRCONIO    ;Código de función
004D 1EFF               MVI     E,0FFH          ;Indicar entrada de consola
004F C30500             JMP     BDOS            ;BDOS devuelve control a la rutina que ha llamado a CONIN

                CONST:
0052 0E0B               MVI     C,B$CONST       ;Código de función
0054 CD0500             CALL    BDOS
0057 B7                 ORA     A               ;Activación del indicador Z para terminar
0058 C9                 RET
```

Figura 5-11. (Continuación.)

Función 9: Visualización de cadena terminada en "$"

Código de función: C=09H
Parámetros de entrada: DE=Dirección del primer byte de la cadena
Parámetros de salida: Ninguno

Ejemplo

```
0009 =          B$PRINTS        EQU     9       ;Impresión de cadena terminada por $
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

000D =          CR              EQU     0DH     ;Retorno de carro
000A =          LF              EQU     0AH     ;Final de línea
0009 =          TAB             EQU     09H     ;Tab. horizontal
0000 0D0A095468MESSAGE:         DB      CR,LF,TAB,'This is a message',CR,LF,'$'

0017 0E09               MVI     C,B$PRINTS      ;Código de función
0019 110000             LXI     D,MESSAGE       ;Apuntador a mensaje
001C CD0500             CALL    BDOS
```

Propósito Esta función envía una cadena de caracteres al dispositivo de consola. La dirección de esta cadena está en los registros DE. Hay que asegurarse de que el último carácter de la cadena es "$"; el BDOS utiliza este carácter como una indicación de final de cadena. El "$" no se envía a la consola.

Mientras el BDOS envía la cadena, expande tabuladores como se ha descrito previamente, controla si hay algún carácter de entrada y controla el CONTROL-S (XOFF, que detiene la salida hasta que ha entrado otro carácter) o CONTROL-P (que conecta o desconecta la repetición de los caracteres de la consola a la impresora).

Notas Uno de los mayores inconvenientes de esta función es el uso de "$" como un carácter de terminación. Como resultado de ello, no se puede enviar una cadena que contenga un "$". Para ser verdaderamente de propósito general, sería mejor utilizar una subrutina que usara el carácter ASCII NUL (00H) como terminal y que simplemente hiciera llamadas

repetitivas a la función CONOUT del BDOS (código 2). La figura 5-3 es un ejemplo de dicha subrutina.

La figura 5-12 muestra un ejemplo de una subrutina que envía uno de varios mensajes. Selecciona el mensaje basado en un código de mensaje que se le da como parámetro. Por consiguiente, es útil para tratar mensajes de error; el código de llamada puede pasarlo a un código de error de 8 bits. Será más flexible convertir esta subrutina que trata mensajes acabados en bytes 00H utilizando las técnicas que se muestran en la figura 5-3.

```
                        ;OM (Salida de mensaje)
                        ;Esta subrutina selecciona un mensaje entre varios en base
                        ; al contenido del registro A como entrada. Entonces permite
                        ; visualizar el mensaje en la consola.

                        ;Cada mensaje se compone con un "$" como último carácter.
                        ; Si el registro A contiene un valor mayor que el número de
                        ; mensajes declarados, OM enviará a la salida "Mensaje desconocido".

                        ;Como opción, OM puede enviar un retorno de carro / salto de línea
                        ; antes de mandar a la salida el texto del mensaje.

                        ;Parámetros de entrada
                        ;       HL -> tabla de mensajes
                        ;                tiene la forma :
                        ;                DB      3       ;Número de mensajes en la tabla
                        ;                DW      MSG0    ;Dirección del texto (A = 0)
                        ;                DW      MSG1    ;(A = 1)
                        ;                DW      MSG2    ;(A = 2)
                        ;
                        ;       MSG0:    DB       'Texto de los mensajes'
                        ;                          ...etc.
                        ;                A = Código del mensaje (de 0 en adelante)
                        ;                B = Salida CR/LF si es distinto de cero
                        ;       Secuencia de llamada
                        ;                LXI     H,MSG$TABLE
                        ;                LDA     MSGCODE
                        ;                MVI     B,0     ;Supresión CR/LF
                        ;                CALL    OM

0009 =          B$PRINTS        EQU     9       ;Impresión cadena-terminada-por-$
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

000D =          CR              EQU     0DH     ;Retorno de carro
000A =          LF              EQU     0AH     ;Salto de línea

0000 0D0A24     OM$CRLF:        DB      CR,LF,'$'
0003 556E6B6E6FOM$UM:           DB      'Unknown Message$'

                OM:
0013 F5                 PUSH    PSW             ;Almacenamiento del código de mensaje
0014 E5                 PUSH    H               ;Almacenamiento apuntador a tabla mensajes
0015 78                 MOV     A,B             ;Comprobación necesidad de CR/LF
0016 B7                 ORA     A
0017 CA2200             JZ      OM$NOCR         ;No
001A 110000             LXI     D,OM$CRLF       ;Salida de CR/LF
001D 0E09               MVI     C,B$PRINTS
001F CD0500             CALL    BDOS

                OM$NOCR:
0022 E1                 POP     H               ;Recuperación apuntador tabla de mensajes
0023 F1                 POP     PSW             ;Recuperación código del mensaje
0024 BE                 CMP     M               ;Comparación con valor máximo
0025 D23700             JNC     OM$ERR          ;Código de error, no <= máx.
0028 23                 INX     H               ;Ignorar valor máx. en tabla
0029 87                 ADD     A               ;Código mensaje * 2
002A 5F                 MOV     E,A             ;Composición del valor (código * 2)
002B 1600               MVI     D,0
002D 19                 DAD     D               ;HL -> Dirección texto mensaje
```

Figura 5-12. Visualización en la consola de mensajes acabados en $.

```
002E 5E             MOV     E,M             ;Obtención byte LS
002F 23             INX     H               ;HL -> byte MS
0030 56             MOV     D,M             ;Obtención byte MS
                                            ;DE -> texto del mensaje

                OM$PS:                      ;Impresión punto de entrada de la cadena
0031 0E09           MVI     C,B$PRINTS      ;Código de función
0033 CD0500         CALL    BDOS
0036 C9             RET                     ;Retorno al peticionario

                OM$ERR:                     ;Error
0037 110300         LXI     D,OM$UM         ;Apuntar a "Mensaje desconocido"
003A C33100         JMP     OM$PS           ;Impresión de cadena
```

Figura 5-12. (Continuación.)

Función 10: Lectura de cadena de la consola

Código de la función: C = 0AH
Parámetros de entrada: DE = Dirección de la memoria intermedia de cadenas
Parámetros de salida: Memoria intermedia de cadena con los bytes de consola incorporados

Ejemplo

```
000A =          B$READCONS      EQU     10      ;Lectura de cadena de consola
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS
0050 =          BUFLEN          EQU     80      ;Longitud de la memoria intermedia

                BUFFER:                         ;Mem. int. de entrada de la consola
0000 50         BUFMAXCH:       DB      BUFLEN  ;Número máximo de caracteres en la
                                                ; memoria intermedia
0001 00         BUFACTCH:       DB      0       ;Número de caracteres de entrada
0002            BUFCH:          DS      BUFLEN  ;Caracteres en memoria interna

0052 0E0A               MVI     C,B$READCONS    ;Código de función
0054 110000             LXI     D,BUFFER        ;Apuntador a memoria intermedia
0057 CD0500             CALL    BDOS
```

Propósito Esta función lee una cadena de caracteres del dispositivo de consola y los almacena en una memoria intermedia (dirección en DE) que se ha definido previamente. Se pueden editar líneas completas: el operador puede retroceder, borrar la línea y volver a empezar, así como utilizar todas las funciones de control normales. Lo último que se ve en la memoria intermedia es la versión final de la cadena de caracteres que se ha introducido, sin ninguno de los errores o caracteres de control utilizados para realizar la edición de la línea.

La memoria intermedia que define el programador tiene un formato especial. El primer byte dice al BDOS el número máximo de caracteres que puede aceptar. El segundo byte está reservado por el BDOS para indicar al operador cuántos caracteres están situados actualmente en la memoria. Los siguientes bytes contienen los caracteres de la cadena.

La entrada de caracteres cesará bien cuando se introduce un CARRIAGE

RETURN o cuando se ha recibido el número máximo de caracteres especificado en la memoria. El CARRIAGE RETURN no se almacena en la memoria como un carácter —sirve únicamente como terminal.

Si el primer carácter introducido es un CARRIAGE RETURN, entonces el BDOS sitúa el byte de "caracteres entradas" a 0. Si se intenta introducir más del número máximo de caracteres, la cuenta de "caracteres entradas" será la misma que el máximo valor permitido.

Notas

Esta función es útil para aceptar la entrada de consola, especialmente por la posibilidad de editar líneas. Puede utilizarse también para respuestas de un solo carácter, tales como "Y/N" (sí o no), porque el operador puede mecanografiar "Y", retroceder y escribir encima "N". Esto le hace adecuado para programas "indulgentes", tolerantes con los seres humanos que cambian de opinión.

La figura 5-13 muestra un ejemplo de subrutina que utiliza esta función. Acepta la entrada de consola, compara la entrada con una tabla y transfiere el control a la subrutina adecuada. Muchos programas interactivos necesitan hacer esto; aceptan una orden del operador y transfieren entonces el control al procesador de órdenes apropiado para que se ocupe de ella.

Este ejemplo incluye también otras dos subrutinas que son útiles en su propio campo. Una compara las cadenas terminadas por bytes nulos (FSCMP), y la otra convierte las letras minúsculas en mayúsculas (FOLD).

```
; RSA
; Retorno dirección subprocesador
; Esta rutina devuelve una o varias direcciones seleccionadas a partir
;  de una tabla por comparación entre cadenas especificadas en ella
;  y la entrada por teclado. Se utiliza normalmente para dar el control
;  a un subprocesador particular de acuerdo con la especificación
;  dada por operador desde el teclado.
;
; Las comparaciones entre cadenas de caracteres se realizan
;  convirtiendo las minúsculas en mayúsculas.
;
; Si la entrada dada por el operador no coincide con ninguna
;  de las cadenas especificadas, el indicador de acarreo
;  se pone a uno, en caso contrario se pone a cero.

; Parámetros de entrada
;         HL -> tabla de selección de subprocesador
;                 Tiene la forma siguiente:
;                 DW      TEXTO,SUBPROC0
;                 DW      TEXT1,SUBPROC1
;                 DW      0       ;Terminal
;         TEXTO:  DB      'add',0 ;terminado por byte 00H
;         TEXT1:  DB      'subtract',0
;         SUBPROC0:
;                 Programa para procesar la función ADD.
;         SUBPROC1:
;                 Programa para procesar la función SUBSTRACT.

; Parámetros de salida
;         DE -> cadena de entrada del operador (terminada
;               por 00H).
```

Figura 5-13. Lectura de cadenas de consola para opciones de teclado.

```
                ;       Acarreo a cero, HL -> subprocesador.
                ;       Acarreo a uno, HL -> 0000H.

                ;Secuencia de llamada
                ;       LXI     H,SUBPROCTAB    ;Tabla de subprocesadores
                ;       CALL    RSA
                ;       JC      ERROR           ;Indicador a uno sólo si error
                ;       LXI     D,RETURN        ;Simulación de instrucción CALL
                ;       PUSH    D               ;Puesta en la pila de la dirección de retorno
                ;       PCHL                    ;Llamada ("CALL") al subprocesador
                ;       RETURN:

000A =          B$READCONS      EQU     10      ;Lectura en la memoria intermedia de la cadena de consola
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0050 =          RSA$BL          EQU     80      ;Longitud de la memoria intermedia
0000 50         RSA$BUF:        DB      RSA$BL  ;Número máximo de caracteres
0001 00         RSA$ACTC:       DB      0       ;Número real de caracteres
0002            RSA$BUFC:       DS      RSA$BL  ;Caracteres en la memoria intermedia
0052 00                         DB      0       ;terminal de seguridad

                RSA:
0053 2B                 DCX     H               ;Ajuste del apuntador a subprocesador por
0054 2B                 DCX     H               ; parte del programa que sigue
0055 E5                 PUSH    H               ;Cabecera de pila (TOS) -> tabla subprocesador-2
0056 0E0A               MVI     C,B$READCONS    ;Código de función
0058 110000             LXI     D,RSA$BUF       ;DE -> memoria intermedia
005B CD0500             CALL    BDOS            ;Lectura de la cadena entrada por el operador y
                                                ; conversión para que termine en 00H
005E 210100             LXI     H,RSA$ACTC      ;HL -> número de caracteres de entrada reales
0061 5E                 MOV     E,M             ;Obtención del número real de caracteres de entrada
0062 1600               MVI     D,0             ;Composición de este valor en una palabra
0064 23                 INX     H               ;HL -> primer carácter de datos
0065 19                 DAD     D               ;HL -> primer carácter NO USADO en la memoria intermedia
0066 3600               MVI     M,0             ;Composición de la mem. int. de entrada terminada en 00H

                RSA$ML:                         ;Comparación de la entrada con los valores especificados
                                                ; Bucle principal
0068 E1                 POP     H               ;Recuperación del apuntador a tabla subprocesador
0069 23                 INX     H               ;Desplazamiento a la cabecera de la entrada siguiente
006A 23                 INX     H               ;HL -> dirección del texto
006B 5E                 MOV     E,M             ;Obtención de la dirección del texto
006C 23                 INX     H
006D 56                 MOV     D,M             DE -> texto

006E 7A                 MOV     A,D             ;Comprobación de llegada a final de tabla subprocesador
006F B3                 ORA     E
0070 CA8500             JZ      RSA$NFND        ;No coincidencia

0073 23                 INX     H               ;HL -> dirección subprocesador
0074 E5                 PUSH    H               ;Almacenamiento de apuntador a tabla subprocesador
0075 210200             LXI     H,RSA$BUFC      ;HL -> caracteres de entrada
0078 CD8A00             CALL    FSCMP           ;Comparación de cadenas convertidas a mayúsculas
007B C26800             JNZ     RSA$ML          ;No coincidencia, desplazamiento a la entrada siguiente
007E E1                 POP     H               ;Coincidencia, recuperación apuntador subprocesador
007F 5E                 MOV     E,M             ;Obtención dirección verdadera subprocesador
0080 23                 INX     H
0081 56                 MOV     D,M             ;DE -> código subprocesador
0082 EB                 XCHG                    ;HL -> código subprocesador
0083 B7                 ORA     A               ;Acarreo a cero (coincidencia)
0084 C9                 RET

                RSA$NFND:
0085 210000             LXI     H,0             ;Indicación de no coincidencia
0088 37                 STC                     ;Acarreo a uno
0089 C9                 RET

                ;FSCMP
                ;Comparación de cadenas convertidas (minúsculas a mayúsculas)
                ;Esta subrutina compara dos cadenas terminadas por un byte 00H
                ;y vuelve con los indicadores de condición puestos a uno
                ;para indicar coincidencia.

                ;Parámetros de entrada
                ;       DE -> cadena 1
                ;       HL -> cadena 2
```

Figura 5-13. (Continuación.)

```
                ;Parámetros de salida
                ;       Puesta a uno de los indicadores de condición (se basa en una
                ;       comparación carácter por carácter de cadena-1 y cadena-2)

                FSCMP:
008A 1A                 LDAX    D               ;Obtención de carácter de cadena-1
008B CD9E00             CALL    FOLD            ;Conversión a mayúscula
008E F5                 PUSH    PSW             ;Almacenamiento del carácter de cadena-1
008F 7E                 MOV     A,M             ;Obtención de carácter cadena-2
0090 CD9E00             CALL    FOLD            ;Conversión a mayúscula
0093 47                 MOV     B,A             ;Almacenamiento de carácter de cadena-2
0094 F1                 POP     PSW             ;Recuperación carácter cadena-1
0095 B8                 CMP     B               ;Comparación 1-2
0096 C0                 RNZ                     ;Si no son iguales, retorno
0097 B7                 ORA     A               ;Iguales, comprobación final de cadena
0098 C8                 RZ                      ;Sí
0099 13                 INX     D               ;No, actualización apuntador a cadena-1
009A 23                 INX     H               ; y apuntador a cadena-2
009B C38A00             JMP     FSCMP           ;Comprobación de caracteres siguientes

                ;FOLD
                ;Convierte una letra minúscula (a-z) en mayúscula (A-Z)
                ;El carácter a convertir está en A a la entrada y a la salida.

                FOLD:
009E 4F                 MOV     C,A             ;Conservación carácter de entrada
009F 3E60               MVI     A,'a'-1         ;Comprobación necesidad de conversión
00A1 B9                 CMP     C               ;Comparación con carácter de entrada
00A2 D2AF00             JNC     FOLDX           ;No, carácter <= "a"
00A5 3E7A               MVI     A,'z'           ;Comprobación carácter < "z"
00A7 B9                 CMP     C
00A8 DAAF00             JC      FOLDX           ;No, carácter > "z"
00AB 3EDF               MVI     A,0DFH          ;Conversión de carácter
00AD A1                 ANA     C
00AE C9                 RET

                FOLDX:
00AF 79                 MOV     A,C             ;Recuperación carácter de entrada original
00B0 C9                 RET
```

Figura 5-13. (Continuación.)

Función 11: Lectura de estado de la consola

Código de función: C=0BH
Parámetros de entrada: Ninguno
Parámetros de salida: A=00H si no hay ningún byte de datos de entrada
A=00FFH si hay byte de datos de entrada

Ejemplo

```
000B =          B$CONST         EQU     11      ;Obtención estado consola
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E0B               MVI     C,B$CONST       ;Código de función
0002 CD0500             CALL    BDOS            ;A = 00 si no hay carácter a la espera
                                                ;A = 0FFH si hay carácter a la espera
```

Propósito Esta función indica si un carácter de entrada de la consola está esperando para ser procesado. A diferencia de las funciones de entrada de consola, que esperan hasta que haya entrada, esta función controla simplemente y vuelve inmediatamente.

Notas Utilícese esta función siempre que se desee interrumpir un programa en ejecución si un carácter del teclado de la consola ha entrado. Solamente es preciso poner una llamada al estado de la consola en el bucle principal del programa. Entonces, si el programa detecta qué datos del teclado están esperando, puede llevar a cabo las acciones necesarias. Normalmente éstas consistirán en saltar al lugar 0000H, abortando el programa en curso e iniciando un arranque caliente.

La figura 5-11 es un ejemplo de subrutina que muestra cómo utilizar esta función.

Función 12: Obtención del número de versión del CP/M

Código de función: C=0CH
Parámetros de entrada: Ninguno
Parámetros de salida: HL=Número de versión

Ejemplo

```
000C =          B$GETVER        EQU     12      ;Obtención número de versión CP/M
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E0C               MVI     C,B$GETVER      ;Código de función
0002 CD0500             CALL    BDOS            ;H = 00 para CP/M
                                                ;L = versión (por ejemplo, 22H para 2.2)
```

Propósito Esta función indica qué versión del CP/M está funcionando en ese momento. Devuelve un valor de dos bytes:

H= 00H para CP/M, H=01H para MP/M.

L = 00H para todas las versiones anteriores al CP/M 2.0.

L = 20H para CP/M 2.0, 21H para 2.1, 22H para 2.2, y así sucesivamente para las versiones siguientes.

Esta información es de interés únicamente si el programa utiliza en su interior alguna lógica dependiente de la versión específica. Por ejemplo, la versión 1.4 del CP/M no soporta las mismas operaciones de entrada/salida de ficheros aleatorios que el CP/M 2.2. Por consiguiente, si el programa usa entrada/salida aleatoria, hay que situar este control al principio para asegurar que está funcionando con la versión adecuada del CP/M.

Notas La figura 5-14 es una subrutina que controla el número de versión de CP/M en curso y, si no es el CP/M 2.2, compone un mensaje aclaratorio en la consola y produce un arranque caliente saltando a la posición 0000H.

```
                        ;CCPM
                        ;Comprobación de si CP/M
                        ;Esta subrutina determina el número de la versión del
                        ;sistema operativo y, si no se trata de la versión 2 de CP/M,
                        ;lanza un mensaje de error y ejecuta un arranque caliente.

                        ;Parámetros de entrada y salida
                        ;       Ninguno

                        ;Secuencia de llamada
                        ;       CALL    CCPM    ;Arranque caliente si no se trata de CP/M 2

0009 =          B$PRINTS        EQU     9       ;Visualiza una cadena terminada en $
000C =          B$GETVER        EQU     12      ;Obtención del número de versión
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

000D =          CR              EQU     0DH     ;Retorno de carro
000A =          LF              EQU     0AH     ;Salto de línea

0000 0D0A       CCPMM:  DB      CR,LF
0002 5468697320         DB      'This program can only run under CP/M version 2.'
0031 0D0A24             DB      CR,LF,'$'

                CCPM:
0034 0E0C               MVI     C,B$GETVER      ;Obtención del número de versión
0036 CD0500             CALL    BDOS
0039 7C                 MOV     A,H             ;H debe ser 0 para CP/M
003A B7                 ORA     A
003B C24700             JNZ     CCPME           ;Debe ser MP/M
003E 7D                 MOV     A,L             ;L = número de versión
003F E6F0               ANI     0F0H            ;Número de versión en el cuarteto MS
0041 FE20               CPI     20H             ;Comprobación de versión 2
0043 C24700             JNZ     CCPME           ;Debe ser una versión primitiva
0046 C9                 RET                     ;Sí, CP/M versión 2

                CCPME:                          ;Error
0047 0E09               MVI     C,B$PRINTS      ;Lanza mensaje de error
0049 110000             LXI     D,CCPMM
004C CD0500             CALL    BDOS
004F C30000             JMP     0               ;Arranque caliente
```

Figura 5-14. Determinación del número de versión del CP/M.

Función 13: Puesta a cero del sistema de disco

Código de función: C=0DH
Parámetros de entrada: Ninguno
Parámetros de salida: Ninguno

Ejemplo

```
000D =          B$DSKRESET      EQU     13      ;Puesta a cero del sistema de disco
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E0D               MVI     C,B$DSKRESET    ;Código de función
0002 CD0500             CALL    BDOS
```

Propósito Esta función requiere al CP/M para que ponga a cero completamente el sistema de ficheros en disco. El CP/M pone a cero entonces sus tablas internas, selecciona el disco lógico A como disco por defecto, pone a cero la dirección DMA en el 0080H (la dirección de la memoria intermedia utilizada por el BDOS para leer y escribir en el disco) y pone todos los discos lógicos en modo de lectura/escritura.

El BDOS tendrá entonces que entrar en todos y cada uno de los discos lógicos. Esto supone leer el directorio de ficheros completo para el disco y reconstruir los vectores de situación (que mantienen la información de qué bloques están libres y cuáles se han utilizado para almacenar ficheros).

Notas Esta función permite cambiar los disquetes bajo control del programa. Si el operador desea simplemente cambiar disquetes, sin que el CP/M se aperciba de ello, el próximo acceso al (ahora diferente) disquete forzará al CP/M a declarar el disco de sólo lectura, impidiendo cualquier otro intento de escribir en él. Si se necesita poner a cero uno o dos discos, mejor que todo el sistema de discos, consúltese la función de puesta a cero del disco (código 37) descrita al final de este capítulo.

La figura 5-15 muestra una subrutina simple que envía un mensaje a la consola, solicitando el cambio de disquete de una unidad específica. Realiza entonces una llamada a la función de puesta a cero del disco, para asegurarse de que CP/M entrará en el disquete en el próximo acceso a la unidad.

```
                        ;CDISK
                        ;Cambio de disco
                        ;Esta subrutina lanza un mensaje pidiendo al usuario el cambio
                        ;del disco lógico que especifica, espera a que éste pulse
                        ;un retorno de carro y efectúa entonces una puesta a cero del disco
                        ;y devuelve control a la rutina que efectuó la llamada.

                        ;Parámetros de entrada
                        ;       A = Disco lógico a cambiar (A = 0, B = 1)

                        ;Parámetros de salida
                        ;       Ninguno

                        ;Secuencia de llamada
                        ;       MVI     A,0             ;Cambio unidad A:
                        ;       CALL    CDISK


000D =          B$DSKRESET      EQU     13      ;Código de función de puesta a cero del disco
0009 =          B$PRINTS        EQU     9       ;Impresión de cadena terminada por $
0001 =          B$CONIN         EQU     1       ;Obtención de entrada por consola
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS
000D =          CR              EQU     0DH
000A =          LF              EQU     0AH

0000 0D0A436861CDISKM:          DB      CR,LF,'Change logical disk
0016 00         CDISKD:         DB      0
0017 3A20616E64                 DB      ': and press Carriage Return to continue$'

                CDISK:
003F C640               ADI     'A'-1           ;Conversión a letra
0041 321600             STA     CDISKD          ;Almacenamiento en mensaje
0044 0E09               MVI     C,B$PRINTS      ;Visualización de mensaje
0046 110000             LXI     D,CDISKM
0049 CD0500             CALL    BDOS

                CDISKW:
004C 0E01               MVI     C,B$CONIN       ;Obtención carácter del teclado
004E CD0500             CALL    BDOS
0051 FE0D               CPI     CR
0053 C24C00             JNZ     CDISKW
0056 0E0D               MVI     C,B$DSKRESET    ;Puesta a cero del sistema de disco
0058 CD0500             CALL    BDOS
005B C9                 RET
```

Figura 5-15. Puesta a cero de la unidad de disco solicitada.

Función 14: Selección del disco lógico

Código de función: C=0EH
Parámetros de entrada: E=Código del disco lógico
00H=Unidad A
01H=Unidad B, etc.
Parámetros de salida: Ninguno

Ejemplo

```
000E =          B$SELDSK        EQU     14      ;Selección del disco lógico
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E0E               MVI     C,B$SELDSK      ;Código de función
0002 1E00               MVI     E,0             ;E = 0 para A:, 1 para B:, etc.
0004 CD0500             CALL    BDOS
```

Propósito Esta función hace tomar como disco lógico por defecto al nombrado en el registro E. Todas las referencias siguientes a los ficheros de disco que no especifiquen el disco utilizarán éste.

Cuando se haga referencia a un fichero de disco que *tenga* un disco lógico explícito en su nombre, no es necesario utilizar de nuevo la función de selección de disco; el BDOS cuidará de ello.

Notas Obsérvese el modo en que el disco está especificado en el registro E. No es el mismo que la especificación de la unidad de disco en el primer byte del bloque de control de fichero. En el FCB, un valor 00H se utiliza para indicar "úsese el disco en curso como disco por defecto" (como se ha especificado en la última llamada de selección de disco o por el operador de la consola). Con esta función, un valor 00H en el registro A significa que A es la unidad seleccionada, un valor 01H significa unidad B, etcétera, hasta 0FH para la unidad P, permitiendo 16 unidades en el sistema.

Si selecciona un disco lógico que no existe en la computadora, el BDOS compondrá el siguiente mensaje:

```
BDOS Err on J: Select
```

Si se mecanografía un CARRIAGE RETURN con el fin de que proceda el programa, el BDOS producirá un arranque caliente y devolverá el control al CCP. Para evitarlo, deberá confiarse en que el operador de la computadora no especifique discos no existentes o construya el programa de tal manera que conozca cuantas unidades de discos lógicas hay en el sistema.

Otro problema relacionado con esta función es que no se puede distinguir un disco lógico para el cual se han construido en el BIOS las tablas adecuadas, pero para el cual no existe ninguna unidad física. El BDOS no comprueba si la unidad está físicamente presente cuando se hace

la llamada de selección de disco. Simplemente coloca algunos valores internos de forma que estén dispuestos para acceder al disco lógico. Si se intenta entonces acceder a una unidad no existente, el BIOS detectará el error. Lo que ocurra a continuación está totalmente fuera del control del BIOS. Los BIOS estándar devolverán el control al BDOS, indicando una condición de error y el BDOS enviará el mensaje

```
BDOS Err on C: Bad Sector
```

Entonces existe una alternativa. Se puede pulsar CARRIAGE RETURN, caso en el cual el BDOS ignorará el error e intentará continuar con lo que parezca que ha sido leído, o bien se puede introducir un CONTROL-C, ocasionando el aborto del programa y obligando al CP/M a realizar un arranque caliente.

Obsérvese que la función de selección de disco no devuelve ningún valor. Si el programa obtiene de nuevo el control, puede suponerse que el disco lógico que se pidió tiene al menos tablas declaradas para ello.

Función 15: Apertura de fichero

Código de función: C=0FH
Parámetros de entrada: DE=Dirección del bloque de control de fichero
Parámetros de salida: A=Código de directorio

Ejemplo

```
000F =          B$OPEN          EQU     15      ;Abrir fichero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda en unidad de disco por defecto
0001 46494C454EFCB$NAME:        DB      'FILENAME'      ;Nombre de fichero
0009 545950     FCB$TYP:        DB      'TYP'   ;Tipo de fichero
000C 00         FCB$EXTENT:     DB      0       ;Extensión
000D 0000       FCB$RESV:       DB      0,0     ;Reservado para CP/M
000F 00         FCB$RECUSED:    DB      0       ;Registros utilizados en esta extensión
0010 0000000000FCB$ABUSED:      DB      0,0,0,0,0,0,0,0  ;Bloques utilizados
0018 0000000000                 DB      0,0,0,0,0,0,0,0
0020 00         FCB$SEQREC:     DB      0       ;Reg. de lectura/escritura secuencial
0021 0000       FCB$RANREC:     DW      0       ;Reg. de lectura/escritura aleatoria
0023 00         FCB$RANRECO:    DB      0       ;Byte de desbordamiento de registro
                                                ; aleatorio (MS)

0024 0E0F               MVI     C,B$OPEN        ;Código de función
0026 110000             LXI     D,FCB           ;DE -> bloque de control de fichero
0029 CD0500             CALL    BDOS            ;A = 0FFH si no se encuentra el fichero
```

Propósito Esta función abre un fichero específico para lectura o escritura. El FCB, cuya dirección deberá estar en el registro DE, indica al CP/M el número de usuario, el disco lógico y el nombre y tipo de fichero. Todos los bytes restantes del FCB se pondrán normalmente a 0.

El código devuelto por el BDOS al registro A indica si el fichero ha sido abierto satisfactoriamente. Si A contiene 0FFH, entonces el BDOS habrá sido incapaz de encontrar la entrada del directorio correcta. Si A=0, 1, 2 o 3, entonces el fichero habrá sido abierto.

Notas

La función de apertura de fichero busca el directorio de ficheros completo en el disco lógico especificado, buscando el nombre, tipo y extensión del especificado en el FCB; es decir, busca una comparación exacta para los bytes 1 a 14 inclusive del FCB. El nombre y tipo de fichero pueden ser ambiguos; es decir, pueden contener caracteres "?". En este caso, el BDOS abrirá el primer fichero del directorio, que comparará con el nombre ambiguo en el FCB. Si el nombre o tipo de fichero es más corto de ocho o tres caracteres respectivamente, entonces los restantes caracteres deberán rellenarse con espacios.

Cuando el BDOS busca el directorio de ficheros, espera encontrar una correspondencia *exacta* con cada carácter del nombre y tipo de fichero, incluidas las letras minúsculas o los caracteres no gráficos. Sin embargo, el BDOS usa sólo los siete bits menos significativos de cada carácter —el bit más significativo se utiliza para indicar características de estado especial, o *atributos* del fichero.

Comparando la extensión del fichero, así como su nombre y tipo, se puede, si se desea, abrirlo en algún punto distinto al de su comienzo. Para el acceso secuencial normal no se deseará esto normalmente, pero si el programa puede predecir la extensión del fichero que se requiere, este es el método de desplazarse directamente a él.

También es posible abrir el mismo fichero más de una vez. Cada caso requiere un FCB aparte. El BDOS no está enterado de lo que está ocurriendo. Realmente sólo puede hacerse cuando se lee el fichero. Cada FCB puede utilizarse para leer el fichero independientemente.

Una vez se ha encontrado el fichero en el directorio, el número de registros y los bloques utilizados se copian de la entrada del directorio en el FCB (bytes 16 al 31 inclusive). Si el fichero ha de ser accesible secuencialmente desde el principio, el registro adecuado (byte 32) deberá ser puesto a cero por el programa.

El valor devuelto al registro A es el número de entrada relativa al directorio de la entrada que compara el FCB. Como se ha explicado previamente, la memoria intermedia que usa el CP/M mantiene una información de 128 bytes del directorio con cuatro entradas del directorio numeradas 0, 1, 2 y 3. Este *código de directorio* es devuelto por casi todas las funciones del BDOS relacionadas con el fichero, pero en circunstancias normales esto interesará únicamente para saber si el valor devuelto a A es 0FFH o no.

La figura 5-16 muestra una subrutina que toma una cadena de caracteres terminada por un byte 00H, crea un FCB válido y abre entonces

el fichero especificado. Se muestra, como parte de este ejemplo, la subrutina BF (construcción FCB). Lleva a cabo el trabajo de mayor dificultad de conversión de una cadena de caracteres ASCII en nombre de fichero y tipo al estilo del FCB.

```
                        ;OPENF
                        ;Abrir fichero

                        ;Dado un apuntador a un nombre de fichero terminado
                        ;en un byte 00H y un área que puede ser usada para el bloque
                        ;de control de fichero, esta subrutina construye un bloque
                        ;de control válido e intenta abrir el fichero.

                        ;Si el fichero ha sido abierto, devuelve el control con el indicador
                        ;de acarreo a cero. Si no lo ha conseguido, devuelve
                        ;el control con la marca a uno.

                        ; Parámetros de entrada
                        ;       DE -> area de 36 bytes para bloque de control de fichero
                        ;       HL -> nombre de fichero terminado por byte 00H de la
                        ;             forma {disco:} Nombre {.tip}
                        ;             (disco y tipo son opcionales)

                        ; Parámetros de salida
                        ;       Indicador de acarreo a cero : Fichero abierto correctamente
                        ;       Indicador de acarreo a uno  : Fichero no abierto

                        ; Secuencia de llamada
                        ;       LXI     D,FCB
                        ;       LXI     H,FNAME
                        ;       CALL    OPENF
                        ;       JC      ERROR
                        ;donde
                        ;FCB:   DS      36              ;Lugar para bloque de control
                        ;FNAME: DB      'A:TESTFILE.DAT',0

000F =                  B$OPEN          EQU     15      ;Código de función de apertura de fichero
0005 =                  BDOS            EQU     5       ;Punto de entrada al BDOS

                        OPENF:
0000 D5                         PUSH    D               ;Conservación de apuntador al FCB
0001 CD0C00                     CALL    BF              ;Construcción del bloque de control
0004 0E0F                       MVI     C,B$OPEN
0006 D1                         POP     D               ;Recuperación apuntador al FCB
0007 CD0500                     CALL    BDOS
000A 17                         RAL                     ;Si A = 0FFH indicador de acarreo a 1
                                                        ;en caso contrario indicador de acarreo a cero
000B C9                         RET

                        ;BF
                        ;Construcción bloque de control de fichero
                        ;Esta subrutina compone en el FCB una cadena terminada
                        ;por un byte 00H (presumiblemente un nombre de fichero),
                        ;fijando el disco, nombre y tipo de fichero y poniendo a cero
                        ;el resto del FCB

                        ;Parámetros de entrada
                        ;       DE -> bloque de control de fichero (36 bytes)
                        ;       HL -> cadena nombre de fichero (terminada por byte 00H)

                        ;Parámetros de salida
                        ;       El bloque de control de fichero construido
                        ;Secuencia de llamada
                        ;       LXI     D,FCB
                        ;       LXI     H,FILENAME
                        ;       CALL    BF

                        BF:
000C 23                         INX     H               ;Comprobación de si segundo carácter es ":"
000D 7E                         MOV     A,M             ;Obtención de carácter de nombre de fichero
000E 2B                         DCX     H               ;HL -> vuelta atrás a primer carácter
000F FE3A                       CPI     ':'             ;Si ":", disco especificado
0011 C21C00                     JNZ     BF$ND           ;Disco no especificado
```

Figura 5-16. Petición de apertura de fichero.

```
0014 7E                 MOV     A,M             ;Obtención letra de disco
0015 E61F               ANI     0001$1111B      ;A (41H) -> 1, B (42H) -> 2 ...
0017 23                 INX     H               ;Ignorar letra del disco
0018 23                 INX     H               ;Se salta ":"
0019 C31D00             JMP     BF$SD           ;Almacenar disco en FCB

                BF$ND:                          ;Disco no presente
001C AF                 XRA     A               ;Indicar disco por defecto

                BF$SD:
001D 12                 STAX    D               ;Almacenar disco en FCB
001E 13                 INX     D               ;DE -> primer carácter del nombre en el FCB
001F 0E08               MVI     C,8             ;Longitud del nombre
0021 CD3700             CALL    BF$GT           ;Obtención de "token"
                                                ;Nota -- en este punto, BF$GT
                                                ;habrá adelantado el apuntador
                                                ;a la cadena hasta encontrar
                                                ;un "." o un byte 00H
0024 FE2E               CPI     '.'             ;Comprobación de carácter terminal
0026 C22A00             JNZ     BF$NT           ;Tipo de fichero no especificado
0029 23                 INX     H               ;Se trata "." en nombre de fichero

                BF$NT:
002A 0E03               MVI     C,3             ;Longitud del tipo de fichero
002C CD3700             CALL    BF$GT           ;Obtención de "token"
                                                ;Nota -- si el tipo de fichero
                                                ;no está presente, BF$GT
                                                ;llenará con espacios el FCB
002F 0600               MVI     B,0             ;Llenado con ceros del resto del FCB
0031 0E18               MVI     C,24            ;36-12 (disco, nombre, tipo = 12 caracteres)
0033 CD6400             CALL    BF$FT           ;Reutilización "token" S/R
0036 C9                 RET

                ;BF$GT
                ;Construcción del FCB -- obtención del "token"

                ;Esta subrutina examina una cadena nombre-de-fichero
                ;poniendo sus caracteres en un bloque de control de fichero.
                ;Rellena con espacios el resto del "token" después de haber
                ;encontrado un carácter terminal ("." o 00H).
                ;Si se encuentra un "*", rellena el resto
                ;del "token" con "?".

                ;Parámetros de entrada
                ;       DE -> en el bloque de control de fichero
                ;       HL -> en la cadena nombre-de-fichero
                ;       C = número máximo de caracteres en el "token"

                ;Parámetros de salida
                ;       El bloque de control de fichero contiene el token siguiente
                ;       A = Carácter de finalización

                BF$GT:
0037 7E                 MOV     A,M             ;Obtención de la cadena de caracteres siguientes
0038 B7                 ORA     A               ;Comprobación de final de cadena
0039 CA5700             JZ      BF$SFT          ;Sí, cadena llenada de espacios
003C FE2A               CPI     '*'             ;Comprobación de necesidad de llenar con "?"
003E CA5C00             JZ      BF$QFT          ;Sí, rellenar con ?
0041 FE2E               CPI     '.'             ;Se supone que el "token" en curso es el
                                                ;nombre de fichero
                                                ;Comprobación de si el tipo de fichero le sigue
                                                ;(Si el token en curso es el tipo
                                                ;de fichero, esta comprobación
                                                ;es redundante)
0043 CA5700             JZ      BF$SFT          ;Sí, rellenar token con espacios
0046 12                 STAX    D               ;Ninguna de las anteriores, almacenar
                                                ;en el FCB
0047 13                 INX     D               ;Actualización del apuntador a FCB
0048 23                 INX     H               ;Actualización del apuntador a cadena
0049 0D                 DCR     C               ;Decrementar longitud de token
004A C23700             JNZ     BF$GT           ;Existen todavía caracteres

                BF$SKIP:                        ;Pasar por alto los caracteres hasta "." o 00H
004D 7E                 MOV     A,M             ;Obtención cadena de caracteres siguiente
004E B7                 ORA     A               ;Comprobación de 00H
```

Figura 5-16. (Continuación.)

```
004F C8              RZ                  ;Si
0050 FE2E            CPI     '.'         ;Comprobación de "."
0052 C8              RZ                  ;Si
0053 23              INX     H           ;Actualización del apuntador a cadena (únicamente)
0054 C34D00          JMP     BF$SKIP     ;Probar carácter siguiente

                BF$SFT:                  ;Rellenar token con espacios
0057 0620            MVI     B,' '
0059 C36400          JMP     BF$FT       ;Código común de relleno de token
                                         ;BF$FT devuelve control a la rutina de llamada

                BF$QFT:                  ;Rellenar token con signo de interrogación
005C 063F            MVI     B,'?'
005E CD6400          CALL    BF$FT       ;Código común de relleno de token
0061 C34D00          JMP     BF$SKIP     ;Pasar por alto "*" múltiples, etc.

                BF$FT:                   ;Relleno de token
0064 F5              PUSH    PSW         ;Almacenamiento carácter de finalización
0065 78              MOV     A,B         ;Obtención de carácter de relleno

                BF$FTL:                  ;Bucle interno
0066 12              STAX    D           ;Almacenar en FCB
0067 13              INX     D           ;Actualización de apuntador al FCB
0068 0D              DCR     C           ;Descontar cuenta residual
0069 C26600          JNZ     BF$FTL      ;Seguir
006C F1              POP     PSW         ;Recuperar carácter de terminación
006D C9              RET
```

Figura 5-16. (Continuación.)

Función 16: Cerrado de fichero

Código de función: C=10H
Parámetros de entrada: DE=Dirección del bloque de control de fichero
Parámetros de salida: A=Código de directorio

Ejemplo

```
0010 =          B$CLOSE         EQU     16      ;Cerrar fichero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000            FCB:            DS      36      ;Bloque de control de fichero

0024 0E10               MVI     C,B$CLOSE       ;Código de función
0026 110000             LXI     D,FCB           ;DE -> bloque de control de fichero
0029 CD0500             CALL    BDOS            ;A = 0,1,2,3 si tiene éxito
                                                ;A = 0FFH si el nombre del fichero
                                                ; no está en el directorio
```

Propósito Esta función termina el proceso de un fichero en el que se ha escrito información. Con el CP/M no es necesario cerrar un fichero que se haya estado leyendo. Sin embargo, si se quiere que el programa funcione correctamente con el MP/M (la versión para múltiples usuarios del CP/M), deberán cerrarse todos los ficheros sin tener en cuenta su uso.

La función de cierre de fichero, igual que la de apertura, devuelve el código de directorio al registro A. El registro A contendrá 0FFH si el BDOS no puede cerrar el fichero satisfactoriamente. Si A es 0, 1, 2 ó 3, entonces éste ha sido cerrado.

Notas Cuando el BDOS cierra un fichero en el que se han escrito datos, escribe los contenidos del FCB en el directorio del disco, actualizando una entrada de directorio ya existente mediante la comparación de disco, nombre, tipo y número de extensión, de la misma forma que lo hace la función de apertura de ficheros.

Obsérvese que el BDOS no transfiere el último registro del fichero al disco durante la operación de cerrado. Unicamente actualiza el directorio. Es preciso que se haga llegar explícitamente al disco cualquier grabación parcial. Si el fichero que se ha creado es de texto ASCII estándar de CP/M, es preciso ocuparse de rellenar la porción no utilizada del registro con los caracteres de final de fichero estándar 1AH, como lo espera el CP/M y según se explicará en la sección dedicada a la función de escritura secuencial (código 21).

Función 17: Búsqueda del primer nombre que corresponda

Código de función: C=11H
Parámetros de entrada: DE=Dirección del bloque de control de fichero
Parámetros de salida: A=Código de directorio

Ejemplo

```
0011 =          B$SEARCHF       EQU     17      ;Búsqueda del primero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda en la unidad de disco por defecto
0001 46494C453FFCB$NAME:        DB      'FILE????'      ;Nombre de fichero ambiguo
0009 543F50     FCB$TYP:        DB      'T?P' ;Tipo de fichero ambiguo
000C 00         FCB$EXTENT:     DB      0       ;Extensión
000D 0000       FCB$RESV:       DB      0,0     ;Reservado para CP/M
000F 00         FCB$RECUSED:    DB      0       ;Registros usados en esta extensión
0010 0000000000FCB$ABUSED:      DB      0,0,0,0,0,0,0,0 ;Bloques usados
0018 0000000000                 DB      0,0,0,0,0,0,0,0
0020 00         FCB$SEQREC:     DB      0       ;Registro de lectura/escritura secuencial
0021 0000       FCB$RANREC:     DW      0       ;Registro de lectura/escritura aleatoria
0023 00         FCB$RANRECO:    DB      0       ;Byte de desbordamiento de registro
                                                ; aleatorio (MS)

0024 0E11               MVI     C,B$SEARCHF     ;Código de función
0026 110000             LXI     D,FCB           ;DE -> bloque de control de fichero
0029 CD0500             CALL    BDOS            ;A = 0,1,2,3.
                                                ;(A * 32) + DMA -> entrada al
                                                ; directorio
                                                ;A = OFFH si no está el nombre
                                                ; del fichero
```

Propósito Esta función examina el directorio hasta la primera entrada, que se corresponde con el nombre, tipo y extensión contenidos en el FCB direccionado por DE. El nombre, tipo y extensión del fichero pueden contener un "?" (ASCII 3FH) en una o más posiciones de carácter. Cuando se presenta un "?", el BDOS lo comparará con *cualquier* carácter del directorio en la posición correspondiente. Esto se conoce como comparación de nombre de fichero ambiguo.

El primer byte de un FCB contiene normalmente el número de código del disco lógico. Un valor 0 indica el disco por defecto, mientras que 1 significa disco A, 2 es B, etc., hasta llegar a un posible máximo de 16 para el disco P. Sin embargo, si este byte contiene un "?", el BDOS buscará el disco lógico por defecto y comparará el nombre y tipo de fichero sin tener en cuenta el número de usuario. Esta función se utiliza normalmente en conjunción con la función de búsqueda del siguiente (que se describe inmediatamente después de esta función). La primera búsqueda, en el proceso de comparación, deja ciertas variables fijas en el BDOS para la utilización de la función de búsqueda del siguiente.

Ambas funciones, búsqueda del primer caso y del siguiente, devuelven un código de directorio en el registro A. En la búsqueda del primer caso, A=0FFH si ningún fichero corresponde al FCB, en caso contrario A=0, 1, 2 ó 3.

Notas Para localizar la entrada particular del directorio que empareje con la función de búsqueda del primero o con la búsqueda del próximo, se multiplica el código de directorio devuelto en A por la longitud de la entrada del directorio (32 bytes). Esto se hace fácilmente añadiendo el registro A a sí mismo cinco veces (véase el programa en la figura 5-17 junto a la etiqueta GNFC). Entonces se añade la dirección de DMA para obtener la dirección práctica en la que se encuentra almacenada la entrada del directorio que se ha comparado.

Existen muchas ocasiones en las que puede necesitarse escribir un programa que acepte un nombre de fichero ambiguo y opere con todos los nombres de fichero que se correspondan. (Las órdenes DIR y ERA construidas en el CCP son ejemplos de uso de nombres de fichero ambiguos.) Para hacer esto, se deben usar varias funciones BDOS: la función de puesta de dirección del DMA (código 26, descrita más adelante en este capítulo), esta función (búsqueda del primer caso), y búsqueda del siguiente (código 18). Todo esto aparece en la subrutina que se presenta en la figura 5-17.

```
;GNF
;Esta subrutina devuelve un FCB activado con el primero de los ficheros
;cuyo nombre coincide con el ambiguo, o (si se especifica
;como parámetro de entrada) el nombre del fichero siguiente.

;Nota : Esta subrutina es sensible al contexto.
;       No debe tenerse en proceso más de un nombre
;       de fichero ambiguo.

;>>>   Atención : Esta subrutina cambia la dirección DMA
;>>>              dentro del BDOS
```

Figura 5-17. Llamadas de búsqueda primera/próxima para nombres de fichero ambiguos.

```
                ;Parámetros de entrada
                ;       DE ->   Nombre de fichero posiblemente ambiguo
                ;                  (terminado en un byte 00)
                ;                  (sólo necesario para petición FIRST)
                ;       HL ->  Bloque de control de fichero
                ;       A = 0  : Devuelve el primer nombre de fichero que coincide
                ;         = NZ : Devuelve el nombre de fichero siguiente que coincide

                ;Parámetros de salida
                ;Acarrero a uno : A = FF, no existe nombre de fichero coincidente
                ;            A no = 0FFH, error en el nombre de fichero de entrada
                ;Acarreo a cero : FCB activado con el nombre siguiente
                ;            HL -> Entrada al directorio devuelta por
                ;                  búsqueda del primero/siguiente.

                ;Secuencia de llamadas
                ;       LXI     D,FILENAME
                ;       LXI     H,FCB
                ;       MVI     A,0     ;o MVI A, 1 para NEXT
                ;       CALL    GNF

0011 =          B$SEARCHF       EQU     17      ;Búsqueda del primer nombre de fichero
0012 =          B$SEARCHN       EQU     18      ;Búsqueda del siguiente nombre de fichero
001A =          B$SETDMA        EQU     26      ;Fijación de la dirección DMA
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0080 =          GNFDMA  EQU     80H             ;Dirección DMA por defecto
000D =          GNFSVL  EQU     13              ;Almacenar longitud (número de caracteres a mover)
0024 =          GNFFCL  EQU     36              ;Longitud del bloque de control de fichero
0000            GNFSV:  DS      GNFSVL          ;Area de almacenamiento para nombre/tipo de fichero

                GNF:
000D E5                 PUSH    H               ;Almacenar apuntador FCB
000E D5                 PUSH    D               ;Almacenar apuntador a nombre de fichero
000F F5                 PUSH    PSW             ;Almacenar indicador primero/siguiente

0010 118000             LXI     D,GNFDMA        ;Fijar DMA a dirección conocida
0013 0E1A               MVI     C,B$SETDMA      ;Código de función
0015 CD0500             CALL    BDOS
0018 F1                 POP     PSW             ;Recuperación de indicador primero/siguiente
0019 E1                 POP     H               ;Recuperación apuntador a nombre de fichero
001A D1                 POP     D               ;Recuperación apuntador a FCB
001B D5                 PUSH    D               ;Almacenar nuevamente apuntador a FCB

001C B7                 ORA     A               ;Control de si FIRST o NEXT
001D C23E00             JNZ     GNFN            ;NEXT (siguiente)
0020 CD9300             CALL    BF              ;Construcción del bloque de control de fichero
0023 E1                 POP     H               ;Recuperación del apuntador a FCB (para reajuste de pila)
0024 D8                 RC                      ;Retorno si hay error en nombre de fichero
0025 E5                 PUSH    H               ;Almacenar nuevamente apuntador a FCB

                                                ;Traslado de nombre de fichero al área
                                                ;de almacenamiento
                                                ;HL -> FCB
0026 110000             LXI     D,GNFSV         ;DE -> área de almacenamiento
0029 0E0D               MVI     C,GNFSVL        ;Obtención de la longitud a guardar
002B CD8A00             CALL    MOVE
002E D1                 POP     D               ;Recuperación de apuntador a FCB
002F D5                 PUSH    D               ;Almacenar de nuevo

0030 0E11               MVI     C,B$SEARCHF     ;Búsqueda FIRST (del primero)
0032 CD0500             CALL    BDOS
0035 E1                 POP     H               ;Recuperar apuntador a FCB
0036 FEFF               CPI     0FFH            ;Comprobación de error
0038 CA7D00             JZ      GNFEX           ;Salida de error
003B C35D00             JMP     GNFC            ;Programa común

                GNFN:                           ;Ejecutar una búsqueda del primero
                                                ;para restablecer contacto
                                                ;con el fichero anterior
                                                ;El FCB del usuario contiene todavía
                                                ;el nombre y tipo
003E CD7F00             CALL    GNFZF           ;Rellenado con ceros salvo nombre y tipo
0041 D1                 POP     D               ;Recuperación de la dirección del FCB
0042 D5                 PUSH    D               ;y vuelta a almacenar
0043 0E11               MVI     C,B$SEARCHF     ;Búsqueda del fichero nuevamente
```

Figura 5-17. (Continuación.)

```
0045 CD0500             CALL    BDOS
0048 D1                 POP     D               ;Recuperación del apuntador a FCB
0049 D5                 PUSH    D               ;y vuelta a guardar
004A 210000             LXI     H,GNFSV         ;Transferir el nombre del fichero desde el área
                                                ;de almacenamiento al FCB
004D 0E0D               MVI     C,GNFSVL        ;Longitud del área de almacenamiento
004F CD8A00             CALL    MOVE

0052 0E12               MVI     C,B$SEARCHN     ;Búsqueda del siguiente
0054 CD0500             CALL    BDOS
0057 E1                 POP     H               ;Recuperación de la dirección del FCB
0058 FEFF               CPI     OFFH            ;Comprobación de error
005A CA7D00             JZ      GNFEX           ;Salida de error

                GNFC:
005D E5                 PUSH    H               ;Almacenamiento de la dirección del FCB
005E 87                 ADD     A               ;Multiplicar código de retorno del BDOS * 32
005F 87                 ADD     A               ;* 4
0060 87                 ADD     A               ;* 8
0061 87                 ADD     A               ;* 16
0062 87                 ADD     A               ;* 32
0063 218000             LXI     H,GNFDMA        ;HL -> dirección DMA
0066 5F                 MOV     E,A             ;Componer un valor (código * 32)
                                                ;en DE
0067 1600               MVI     D,0
0069 19                 DAD     D               ;HL -> entrada al directorio de ficheros

                                                ;Transferir nombre de fichero al FCB
006A D1                 POP     D               ;Recuperación de la dirección del FCB
006B E5                 PUSH    H               ;Almacenar apuntador a la entrada al directorio
006C D5                 PUSH    D               ;Almacenarla de nuevo
006D 0E0D               MVI     C,GNFSVL        ;Longitud del área de almacenamiento
006F CD8A00             CALL    MOVE
0072 3A0000             LDA     GNFSV           ;Obtención del disco del área de almacenamiento
0075 D1                 POP     D               ;Recuperación de la dirección del FCB
0076 12                 STAX    D               ;Escritura destructiva sobre el número de usuario en el FCB

                                                ;Poner a ceros la parte final
                                                ;del FCB
0077 CD7F00             CALL    GNFZF           ;Llenado de ceros
007A E1                 POP     H               ;Recuperación apuntador a entrada
                                                ;del directorio
007B AF                 XRA     A               ;Acarreo a cero
007C C9                 RET

                GNFEX:
007D 37                 STC                     ;Acarreo a uno para indicar error
007E C9                 RET

                ;GNFZF
                ;Búsqueda de fichero siguiente -- rellenado de ceros
                ;Esta subrutina rellena con ceros los bytes que siguen
                ;al nombre y tipo de fichero en el FCB

                ;Parámetros de entrada
                ;       DE -> bloque de control de fichero

                GNFZF:
007F 210D00             LXI     H,GNFSVL        ;Salto por encima del área que contiene el nombre de fichero
0082 19                 DAD     D               ;HL -> FCB + GNFSVL
0083 54                 MOV     D,H             ;DE -> FCB + GNFSVL
0084 5D                 MOV     E,L
0085 13                 INX     D               ;DE -> FCB + GNFSVL + 1
0086 3600               MVI     M,0             ;FCB + GNFSVL = 0
0088 0E17               MVI     C,GNFFCL-GNFSVL ;Resto del bloque de control de fichero

                ;Salto a MOVE
                ;Llenado de ceros hasta completar
                ;el FCB

                ;MOVE
                ;Esta subrutina desplaza C bytes desde HL a DE.

                MOVE:
008A 7E                 MOV     A,M             ;Obtención del byte fuente
008B 12                 STAX    D               ;Almacenamiento del byte destino
```

Figura 5-17. (Continuación.)

```
008C 13              INX     D              ;Incremento del apuntador destino
008D 23              INX     H              ;Incremento del apuntador fuente
008E 0D              DCR     C              ;Decremento del contador
008F C28A00          JNZ     MOVE           ;Vuelta atrás a por más
0092 C9              RET

                ;BF
                ;Construcción del bloque de control de fichero

                ;Esta subrutina compone una cadena terminada en un byte 00H
                ;(normalmente un nombre de fichero) en un FCB,
                ;fijando el disco y el nombre y tipo de fichero,
                ;y poniendo a cero el resto del FCB.

                ;Parámetros de entrada
                ;       DE -> bloque de control de fichero (36 bytes)
                ;       HL -> cadena nombre-de-fichero (terminada por un byte 00H)

                ;Parámetros de salida
                ;       El bloque de control de fichero construido

                ;Esta rutina se muestra en forma integra en la figura 5-16

0093 C9         BF:     RET                    ;Subrutina ficticia para este ejemplo
```

Figura 5-17. (Continuación.)

Función 18: Búsqueda del siguiente nombre que corresponda

Código de función: C=12H
Parámetros de entrada: Ninguno (toma la llamada de la búsqueda primera previa)
Parámetros de salida: A=Código del directorio

Ejemplo

```
0012 =          B$SEARCHN       EQU     18      ;Búsqueda del siguiente
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E12               MVI     C,B$SEARCHN     ;Código de función
                                                ;Nota: No apuntador a FCB
                                                ;Esta llamada debe estar precedida por
                                                ; una llamada a búsqueda del primero
0002 CD0500             CALL    BDOS            ;A = 0,1,2,3
                                                ;(A * 32) + DMA -> entrada al
                                                ; directorio
                                                ;A = OFFH si no se encuentra nombre
                                                ; de fichero
```

Propósito Esta función busca en el directorio el *próximo* nombre, tipo y extensión de fichero que se corresponda con el FCB especificado en una llamada previa de la función de búsqueda del primer caso.

La búsqueda del primer caso y la búsqueda del siguiente son las únicas funciones del BDOS que deben utilizarse juntas. Como puede verse, la función de búsqueda del siguiente no requiere una dirección del FCB como parámetro de entrada —toda la información necesaria se habrá dejado en el BDOS en la llamada a la función de búsqueda del primer caso.

Al igual que en el caso anterior, la búsqueda del siguiente caso devuelve

un código de directorio al registro A; en este caso, si A=0FFH, significa que no hay *más* ficheros que se puedan comparar con el bloque de control de fichero. Si A es distinto a 0FFH, tendrá un valor de 0, 1, 2 ó 3, indicando el número de entrada al directorio correspondiente.

Notas

Existen dos formas de utilizar las llamadas de búsqueda primera/siguiente. Consideremos un simple programa de copia de fichero que toma como entrada un nombre de fichero ambiguo. Se puede examinar el directorio, comparando todos los nombres de fichero posibles, quizá haciéndolos aparecer en la consola y almacenando los nombres de los ficheros que han de ser copiados en una tabla dentro del programa. Esto tendría la ventaja de posibilitar la presentación de los nombres de fichero al operador antes de que se realice ninguna copia. También se podría prever que el operador seleccionase los ficheros que se han de copiar fichero a fichero. Un inconveniente sería que no podría predecirse exactamente cuántos ficheros habrían de seleccionarse. En algunos sistemas de disco rígido posiblemente habría que acomodar varios miles de nombres de fichero.

El método alternativo de enfrentarse con el problema sería comparar un nombre, copiarlo y entonces comparar el próximo nombre, copiarlo, etc. Si se da al operador la facultad de elegir los ficheros que se han de copiar, éste tendrá que esperar en el terminal, según se va copiando cada fichero, pero el programa no necesitará tener grandes áreas de tablas para conservar los nombres de ficheros. Esta solución al problema es ligeramente más complicada, según puede verse en la lógica de la figura 5-17.

La subrutina de la figura 5-17, Obtener el próximo fichero (GNF), contiene toda la lógica necesaria para buscar un directorio para ambas alternativas descritas. Requerirá que se indique *al entrar* si debe buscar por el primer caso o por el siguiente, colocando A a cero o en algún valor distinto a cero respectivamente.

Como puede verse en la figura 5-17, cuando se llama a la subrutina para obtener el fichero *siguiente* es preciso ejecutar una función de búsqueda del primer caso para volver a encontrar el fichero previo. Sólo entonces puede lanzarse la búsqueda siguiente.

Como con todas las funciones que devuelven un código de directorio en A, si este valor no es 0FFH, será el número de entrada del directorio correspondiente en el registro de directorio normalmente en memoria. Esta grabación del directorio habrá sido leída en la memoria en aquella dirección especificada en la última llamada a la función de fijación de dirección de DMA (código 26, 1AH). A pesar de su extraño nombre, la dirección DMA es simplemente la dirección en la cual se situará cualquier entrada del disco. Si la función de fijación de la dirección DMA no se ha utilizado para cambiar el valor, entonces la dirección DMA por defecto de CP/M, posición 0080H, se utilizará para contener el registro del directorio.

El código para localizar la dirección de la entrada del directorio particular comparado por las funciones de búsqueda del primer caso/siguiente se muestra en la figura 5-17 cerca de la etiqueta GNFC. El método incluye la multiplicación del código de directorio por 32 y adición de este producto a la dirección DMA actual.

Función 19: Borrado de fichero

Código de función: C=13H
Parámetros de entrada: DE=Dirección del bloque de control de fichero
Parámetros de salida: A=Código de directorio

Ejemplo

```
0013 =          B$ERASE         EQU     19      ;Borrar fichero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda en el disco por defecto
0001 3F3F4C454EFCB$NAME:        DB      '??LENAME'      ;Nombre de fichero ambiguo
0009 3F5950     FCB$TYP:        DB      '?YP'   ;Tipo de fichero ambiguo
000C 00         FCB$EXTENT:     DB      0       ;Extensión
000D 0000       FCB$RESV:       DB      0,0     ;Reservado al CP/M
000F 00         FCB$RECUSED:    DB      0       ;Registros usados en esta extensión
0010 0000000000FCB$ABUSED:      DB      0,0,0,0,0,0,0,0 ;Bloques utilizados
0018 0000000000                 DB      0,0,0,0,0,0,0,0
0020 00         FCB$SEQREC:     DB      0       ;Reg. de lectura/escritura secuencial
0021 0000       FCB$RANREC:     DW      0       ;Reg. de lectura/escritura aleatoria
0023 00         FCB$RANRECO:    DB      0       ;Byte de desbordamiento de
                                                ; registro aleatorio (MS)

0024 0E13               MVI     C,B$ERASE       ;Código de función
0026 110000             LXI     D,FCB           ;DE -> bloque de control de fichero
0029 CD0500             CALL    BDOS            ;A = 0FFH si no se encuentra el fichero
```

Propósito Esta función borra lógicamente del directorio de ficheros aquellos que corresponden al FCB direccionado por DE. Lo hace reemplazando el primer byte de cada entrada relevante del directorio (recuérdese que un solo fichero puede tener varias entradas, una para cada extensión) por el valor 0E5H, lo que hace aparecer la entrada del directorio como disponible para ser usada.

Notas Al igual que las dos funciones precedentes, esta función puede tomar un nombre y tipo de fichero ambiguos como parte del bloque de control de fichero, pero, contrariamente a las otras dos funciones, el código de selección de disco lógico no podrá ser un "?".

Esta función devuelve un código de directorio en A de la misma forma que las operaciones de fichero previas.

Función 20: Lectura secuencial

Código de función: C=14H
Parámetros de entrada: DE=Dirección de bloque de control de fichero
Parámetros de salida: A=Código de directorio

Ejemplo

```
0014 =          B$READSEQ    EQU    20      ;Lectura secuencial
0005 =          BDOS         EQU    5       ;Punto de entrada al BDOS

                FCB:                        ;Bloque de control de fichero
0000 00         FCB$DISK:    DB     0       ;Búsqueda en la unidad de disco por defecto
0001 46494C454EFCB$NAME:     DB     'FILENAME'   ;Nombre de fichero
0009 545950     FCB$TYP:     DB     'TYP' ;Tipo de fichero
000C                         DS     24      ;Fijado por apertura de fichero


                                            ;El registro será leído en la dirección
                                            ; fijada por la llamada anterior
                                            ; a SETDMA
0024 0E14            MVI    C,B$READSEQ     ;Código de función
0026 110000          LXI    D,FCB           ;DE -> bloque de control de fichero
0029 CD0500          CALL   BDOS            ;A = 00H si la operación ha tenido éxito
                                            ;A = distinto de cero si no hay datos
                                            ; en el fichero
```

Propósito Esta función lee el próximo registro (sector de 128 bytes) del fichero designado en la memoria en la dirección indicada por medio de la última llamada a la función de fijación de dirección DMA (código 26, 1AH). El fichero leído está especificado por el campo de registro secuencial del FCB (FCB$SEQREC en el ejemplo de listado para la función de apertura de fichero, código 15). Este campo se incrementa en 1 de forma que la llamada siguiente a lectura secuencial obtenga el siguiente registro del fichero. Si se alcanza el final de la extensión en curso, entonces el BDOS abrirá automáticamente la próxima extensión y pondrá a cero el campo de registro secuencial, dispuesto para la próxima llamada a función de lectura.

El fichero especificado en el FCB debe haber sido preparado para entradas por una llamada de apertura de fichero (código 15, 0FH) o de creación de fichero (código 22, 16H).

El valor 00H es devuelto en A para indicar una operación de lectura secuencial satisfactoria, mientras que un valor distinto a cero demuestra que la lectura no se ha podido completar por no haber datos en el registro siguiente, como al final del fichero.

Notas Aunque no es evidente, se puede cambiar el número de registro secuencial, FCB$SEQREC, y con una extensión dada leer un registro al azar. Si se desea acceder a un registro dado dentro de un fichero, es preciso calcular qué extensión tendrá dicho registro y fijar el campo de extensión en el bloque de control de fichero (FCB$EXTENT) antes de abrirlo. Entonces, aunque el nombre de la función implique un acceso secuencial, en la

práctica se puede usar para hacer un tipo sencillo de acceso aleatorio. Si se necesita realizar un acceso aleatorio real, consúltese más adelante la función de lectura aleatoria (código 33), que se encarga de abrir automáticamente la extensión correcta.

La figura 5-18 muestra un ejemplo de una subrutina que devuelve los datos de fichero secuencial byte a byte leyendo tantos registros del fichero como sean necesarios. Esta subrutina, GETC, es útil como "primitiva" de bajo nivel con la que se pueden construir funciones más sofisticadas, tales como las que leen un número determinado de caracteres o leen caracteres hasta una combinación de CARRIAGE RETURN/LINE FEED.

Cuando se leen datos de un fichero de texto CP/M, lo convencional es rellenar el último registro del fichero con caracteres 1AH (CONTROL-Z). Por tanto, dos condiciones posibles puede indicar el final de fichero: bien si se encuentra un 1AH, o si se recibe un código 0FFH desde la función del BDOS (en el registro A). Sin embargo, si el fichero que se está leyendo no es de texto ASCII, entonces un carácter 1AH no tiene significado especial —es únicamente un byte de datos normal en el cuerpo del fichero.

Función 21: Escritura secuencial

Código de función: C=15H
Parámetros de entrada: DE=Dirección del bloque de control de fichero
Parámetros de salida: A=Código de directorio

Ejemplo

```
0015 =          B$WRITESEQ      EQU     21      ;Escritura secuencial
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda en unidad de disco por defecto
0001 46494C454EFCB$NAME:        DB      'FILENAME'      ;Nombre de fichero
0009 545950     FCB$TYP:        DB      'TYP'   ;Tipo de fichero
000C                            DS      24      ;Activación por "Abrir" o "Crear" fichero

                                                ;El registro debe estar en la dirección
                                                ; fijada por la llamada previa a SETDMA
0024 0E15               MVI     C,B$WRITESEQ    ;Código de función
0026 110000             LXI     D,FCB           ;DE -< bloque de control de fichero
0029 CD0500             CALL    BDOS            ;A = 00H si la operación
                                                ; ha tenido éxito
                                                ;A = distinto de 0 si el disco está lleno
```

Propósito Esta función escribe un registro en una dirección especificada en la última llamada de función que fija la dirección DMA (código 26, 1AH) en el fichero definido en el FCB. El número de registro secuencial del FCB (FCB$SEQREC) se incrementa en 1 de forma que la próxima llamada de escritura secuencial escribirá en el registro que ocupa la posición siguiente en el fichero. Si es necesario, se abrirá una nueva extensión para recibir el nuevo registro.

```
                ; GETC
                ; Esta subrutina obtiene el carácter siguiente
                ; de un fichero secuencial en disco. Supone que
                ; el fichero ha sido ya abierto.

                ;>>>    Nota : Esta subrutina cambia la dirección DMA de CP/M.

                ;Parámetros de entrada
                ;         DE -> bloque de control de fichero

                ;Parámetros de salida
                ;         A = carácter siguiente del fichero
                ;             (= 0FFH en el final físico del fichero)
                ;             Nota : 1AH es el carácter normal de fin de fichero
                ;                    (EOF) en los ficheros ASCII.

                ; Secuencia de llamada
                ;         LXI     DE,FCB
                ;         CALL    GETC
                ;         CPI     1AH
                ;         JZ      EOFCHAR
                ;         CPI     0FFH
                ;         JZ      ACTUALEOF

0014 =          B$READSEQ       EQU     20      ;Lectura secuencial
001A =          B$SETDMA        EQU     26      ;Fija la dirección DMA
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0080 =          GETCBS  EQU     128             ;Tamaño de memoria intermedia
0000            GETCBF: DS      GETCBS          ;Declaración de memoria intermedia
0080 00         GETCCC: DB      0               ;Contador de caracteres (inicialmente
                                                ;"vacío")

                GETC:
0081 3A8000             LDA     GETCCC          ;Comprobación de memoria intermedia vacía
0084 B7                 ORA     A
0085 CA9900             JZ      GETCFB          ;Sí, rellenar memoria intermedia

                GETCRE:                         ;Punto de reentrada después de llena la memoria intermedia
0088 3D                 DCR     A               ;No, decrementar contador
0089 328000             STA     GETCCC          ;Almacenar cuenta decrementada

008C 47                 MOV     B,A             ;Cálculo del desplazamiento del siguiente
                                                ;carácter
008D 3E7F               MVI     A,GETCBS-1      ;Por sustracción
008F 90                 SUB     B               ;(Tamaño memoria intermedia -- cuenta decrementada)
0090 5F                 MOV     E,A             ;Componer el valor sobre palabra
0091 1600               MVI     D,0
0093 210000             LXI     H,GETCBF        ;HL -> base de la memoria de intermedia
0096 19                 DAD     D               ;HL -> carácter siguiente en memoria intermedia
0097 7E                 MOV     A,M             ;Obtención carácter siguiente
0098 C9                 RET

                GETCFB:                         ;Llenado de la memoria intermedia
0099 D5                 PUSH    D               ;Guardar apuntador a FCB
009A 110000             LXI     D,GETCBF        ;Fijar dirección DMA a la memoria intermedia
009D 0E1A               MVI     C,B$SETDMA      ;Código de función
009F CD0500             CALL    BDOS
00A2 D1                 POP     D               ;Recuperación apuntador a FCB
00A3 0E14               MVI     C,B$READSEQ     ;Lectura de "registro" (sector) secuencial
00A5 CD0500             CALL    BDOS
00A8 B7                 ORA     A               ;Comprobación de lectura no posible (A = NZ)
00A9 C2B400             JNZ     GETCX           ;Sí
00AC 3E80               MVI     A,GETCBS        ;Puesta a cero contador
00AE 328000             STA     GETCCC
00B1 C38800             JMP     GETCRE          ;Subrutina de reentrada

                GETCX:                          ;Final físico de fichero
00B4 3EFF               MVI     A,0FFH          ;Indicarlo
00B6 C9                 RET
```

Figura 5-18. Lectura del próximo carácter de un fichero secuencial en disco.

Esta función es análoga a la función de lectura secuencial, escribiendo en lugar de leer. El fichero especificado en el FCB deberá ser activado primero por medio de una apertura de fichero (código 15, 0FH) o por una llamada de creación de fichero (código 22, 16H).

Un código de directorio 00H se devuelve a A para indicar que la escritura se ha realizado; un valor distinto de cero en A significará que la escritura no ha podido completarse por estar lleno el disco.

Notas

Al igual que con la función de lectura secuencial (código 20, 14H), se puede llevar a cabo de forma sencilla la escritura aleatoria en el fichero manipulando el número de registro secuencial (FCB$SEQREC). Sin embargo, sólo puede escribirse sobre registros *existentes* en el fichero, y si se quiere cambiar a otra extensión, es preciso cerrar el fichero y reabrirlo poniendo el valor correcto en el campo FCB$EXTENT. Para escribir realmente de forma aleatoria en el fichero, véase más adelante la función de escritura aleatoria (código 34, 22H). Esta se ocupa de abrir o crear la extensión correcta del fichero automáticamente.

La única condición de error lógica que puede ocurrir cuando se escribe en un fichero es la de tener insuficiente espacio en el disco para acomodar la próxima extensión del fichero. Todos los errores de hardware detectados serán tratados por el controlador de disco construido en el interior del BIOS o del BDOS.

La figura 5-19 muestra la subrutina PUTC, a la cual se pueden pasar datos un byte cada vez. Ensambla estos datos en una memoria intermedia, realizando una llamada a escritura secuencial siempre que ésta se llena. Puede verse que se ha hecho una provisión en los parámetros de entrada (colocando el registro B en un valor distinto de cero) para que la subrutina llene los restantes caracteres no utilizados de la memoria intermedia con caracteres 1AH. Esto debe realizarse para señalar el final de un fichero de texto ASCII.

Función 22: Creación (formación) de un fichero

Código de función: C=16H
Parámetros de entrada: DE=Dirección del bloque de control de fichero
Parámetros de salida: A=Código del directorio

Ejemplo

```
0016 =          B$CREATE        EQU     22      ;Creación de fichero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda en la unidad de disco
                                                ; por defecto
0001 46494C454EFCB$NAME:        DB      'FILENAME'      ;Nombre de fichero
0009 545950     FCB$TYP:        DB      'TYP'   ;Tipo de fichero
000C 00         FCB$EXTENT:     DB      0       ;Extensión
```

```
000D 0000          FCB$RESV:       DB      0,0     ;Reservado para CP/M
000F 00            FCB$RECUSED:    DB      0       ;Registros usados en esta extensión
0010 0000000000FCB$ABUSED:         DB      0,0,0,0,0,0,0,0  ;Bloques utilizados
0018 0000000000                    DB      0,0,0,0,0,0,0,0
0020 00            FCB$SEQREC:     DB      0       ;Reg. de lectura/escritura secuencial
0021 0000          FCB$RANREC:     DW      0       ;Reg. de lectura/escritura aleatoria
0023 00            FCB$RANRECO:    DB      0       ;Byte de desbordamiento de
                                                   ; registro aleatorio (MS)

                                                   ;Nota : El fichero creado no existe
                                                   ;todavía
0024 0E16                  MVI     C,B$CREATE      ;Código de función
0026 110000                LXI     D,FCB           ;DE -> bloque de control de fichero
0029 CD0500                CALL    BDOS            ;A = 0,1,2,3 si la operación
                                                   ; tiene éxito
                                                   ;A = 0FFH si directorio lleno
```

```
                ;PUTC
                ;Esta subrutina sitúa el carácter siguiente al final
                ;de un fichero secuencial, escribiendo en registros completos
                ;(sectores de 128 bytes) o, si se desea, rellenará
                ;el registro en curso no completo con caracteres 1AH
                ;que indican final de fichero a CP/M

                ;Parámetros de entrada
                ;       DE -> bloque de control de fichero
                ;       B = 0, A = carácter de datos siguiente para salida
                ;       B /= 0, distinto de cero rellenar el registro en curso con 1AH

                ;Parámetros de salida
                ;       Ninguno

                ;Secuencia de llamada
                ;       LXI     D,FCB
                ;       MVI     B,0     ;No final de fichero
                ;       LDA     CHAR
                ;       CALL    PUTC
                ;     o
                ;       LXI     D,FCB
                ;       MVI     B,1     ;Indicación de final de fichero
                ;       CALL    PUTC

0015 =          B$WRITESEQ      EQU     21      ;Escritura secuencial
001A =          B$SETDMA        EQU     26      ;Fijar dirección DMA
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0080 =          PUTCBS  EQU     128             ;Tamaño de la memoria intermedia
0000            PUTCBF: DS      PUTCBS          ;Declaración de memoria intermedia
0080 00         PUTCCC: DB      0               ;Contador de caracteres (inic. vacio)

                PUTC:
0081 D5                 PUSH    D               ;Almacenar dirección FCB
0082 F5                 PUSH    PSW             ;Almacenar carácter-dato
0083 78                 MOV     A,B             ;Comprobación de petición de final de fichero
0084 B7                 ORA     A
0085 C29900             JNZ     PUTCEF          ;Sí
0088 CDC300             CALL    PUTCGA          ;No, obtención dirección siguiente byte libre
                                                ;HL -> siguiente byte libre
                                                ;E = cuenta de caracteres en curso (tal
                                                ;como A)
008B F1                 POP     PSW             ;Recuperación carácter-dato
008C 77                 MOV     M,A             ;Almacenamiento en memoria intermedia
008D 7B                 MOV     A,E             ;Obtención cuenta de caracteres en curso
008E 3C                 INR     A               ;Actualización contador caracteres
008F FE80               CPI     PUTCBS          ;Comprobación de memoria intermedia completa
0091 CAA900             JZ      PUTCWB          ;Si, escritura de la memoria intermedia
0094 328000             STA     PUTCCC          ;No, guardar contador actualizado
0097 D1                 POP     D               ;Obtención dirección FCB para retorno
0098 C9                 RET
```

Figura 5-19. Escritura del próximo carácter en un fichero secuencial en disco.

```
                PUTCEF:                         ;Final de fichero
0099 F1                 POP     PSW             ;Obtención carácter-dato
009A CDC300             CALL    PUTCGA          ;HL -> siguiente byte libre
                                                ;A = contador carácter en curso

                PUTCCE:                         ;Copiar carácter EOF (final fichero)
009D FE80               CPI     PUTCBS          ;Comprobación final memoria intermedia
009F CAA900             JZ      PUTCWB          ;Si, escritura de la memoria intermedia
00A2 361A               MVI     M,1AH           ;No, almacenar EOF en la memoria intermedia
00A4 3C                 INR     A               ;Actualización del contador
00A5 23                 INX     H               ;Actualización apuntador a memoria intermedia
00A6 C39D00             JMP     PUTCCE          ;Continúe hasta el final de memoria intermedia

                PUTCWB:                         ;Escritura de la memoria intermedia
00A9 AF                 XRA     A               ;Puesta a 0 del contador de caracteres
00AA 328000             STA     PUTCCC
00AD 110000             LXI     D,PUTCBF        ;DE -> memoria intermedia
00B0 0E1A               MVI     C,B$SETDMA      ;Fijación dirección DMA -> memoria intermedia
00B2 CD0500             CALL    BDOS
00B5 D1                 POP     D               ;Recuperación dirección FCB
00B6 0E15               MVI     C,B$WRITESEQ    ;Escritura de registro secuencial
00B8 CD0500             CALL    BDOS
00BB B7                 ORA     A               ;Comprobación de error
00BC C2C000             JNZ     PUTCX           ;Si, si A = NZ
00BF C9                 RET                     ;No, retorno

                PUTCX:                          ;Salida de error
00C0 3EFF               MVI     A,0FFH          ;Indicación de ello
00C2 C9                 RET

                PUTCGA:                         ;Retorno con HL -> siguiente carácter libre
                                                ;y A = cuenta carácter en curso
00C3 3A8000             LDA     PUTCCC          ;Obtención cuenta de carácter en curso
00C6 5F                 MOV     E,A             ;Puesta en DE del valor
00C7 1600               MVI     D,0
00C9 210000             LXI     H,PUTCBF        ;HL -> base de la memoria intermedia
00CC 19                 DAD     D               ;HL -> siguiente carácter libre
00CD C9                 RET
```

Figura 5-19. (Continuación).

Propósito Esta función crea un nuevo fichero del nombre y tipo especificados. Primero hay que asegurarse de que no existe ya ningún fichero del mismo nombre y tipo en el mismo disco lógico, bien intentando abrirlo (si esto tiene éxito, el fichero existe ya) o borrándolo incondicionalmente.

Además de crear el fichero y su entrada al directorio asociada, esta función abre también efectivamente el fichero, de forma que éste está preparado para que se escriba en él.

Esta función devuelve un código normal de directorio si la creación del fichero se ha completado satisfactoriamente o un valor 00FFH si no existe suficiente espacio en el disco o en el directorio.

Notas En estas circunstancias puede desearse crear un fichero que sea algo más "seguro" que un fichero normal del CP/M. Se puede hacer utilizando bien letras minúsculas o caracteres ASCII no gráficos, tales como ASCII NUL (00H) en el nombre o tipo de fichero. Ninguna de estas clases de caracteres puede generarse desde el teclado; en el primer caso, el CCP cambia todos los caracteres de minúsculas a mayúsculas, y en el segundo rechaza los nombres que posean caracteres raros. Así, el operador no puede borrar estos ficheros porque no existe ninguna forma para crear el mismo nombre desde el CCP.

La inversa es también cierta; la única forma en que se pueden borrar estos ficheros es utilizando un programa que *pueda* fijar el nombre exacto del fichero en un FCB y entonces lanzar una llamada a la función de borrado de fichero.

Obsérvese que esta función no puede aceptar un nombre de fichero ambiguo en el FCB.

La figura 5-20 muestra una subrutina que crea un fichero solamente después de haber borrado todos los existentes del mismo nombre.

Función 23: Cambio de nombre de un fichero

Código de función: C=17H
Parámetros de entrada: DE=Dirección del bloque de control de fichero
Parámetros de salida: A=Código de directorio

Ejemplo

```
0017 =          B$RENAME        EQU     23      ;Cambiar de nombre un fichero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00                         DB      0       ;Búsqueda en unidad de disco por
                                                ; defecto
0001 4F4C444E41                 DB      'OLDNAME'       ;Nombre de fichero
0009 545950                     DB      'TYP'   ;Tipo de fichero
000C 00000000                   DB      0,0,0,0
0010 00                         DB      0       ;FCB + 16
0011 4E45574E41                 DB      'NEWNAME'       ;Nombre de fichero
0019 545950                     DB      'TYP'   ;Tipo de fichero
001C 00000000                   DB      0,0,0,0

0020 0E17               MVI     C,B$RENAME      ;Código de función
0022 110000             LXI     D,FCB           ;DE -> bloque de control de fichero
0025 CD0500             CALL    BDOS            ;A = 00H si éxito en operación
                                                ;A = 0FFH si no se encuentra fichero
```

Propósito Esta función da nuevo nombre y tipo a un fichero existente, proporcionándole, por tanto, nuevo nombre y tipo. No es corriente para ello utilizar un único FCB para almacenar tanto el nombre y tipo antiguos (en los primeros 16 bytes) y el nuevo nombre y tipo (en los segundos 16 bytes).

Esta función devuelve un código de directorio normal si el cambio de nombre se ha completado satisfactoriamente o se tiene un valor 0FFH si el nombre antiguo no se ha podido encontrar.

Notas La función cambio de nombre solamente controla que existan el nombre y tipo antiguos del fichero; no controla para asegurarse que la combinación de nombre y tipo nuevos no exista ya. Por tanto, deberá intentarse abrir el nuevo nombre y tipo del fichero. Si se consigue, no hay que intentar la operación. El CP/M creará más de un fichero del mismo nombre y tipo, y se puede perder la información de ambos si se intenta resolver el problema.

Para mayor seguridad, se pueden utilizar también letras minúsculas y caracteres no gráficos para nombre y tipo de fichero, como se ha descrito anteriormente en la función de creación de fichero (código 22, 16H).

```
                        ;CF
                        ;Creación de fichero
                        ;Esta subrutina crea un fichero borrando cualquier otro
                        ;antes de crear el nuevo.

                        ;Parámetros de entrada
                        ;       DE -> bloque de control de fichero para el nuevo

                        ;Parámetros de salida
                        ;       Acarreo a cero si la operación tiene éxito
                        ;                       (A = 0,1,2,3)
                        ;       Acarreo a uno si hay error (A = OFFH)

                        ;Secuencia de llamada
                        ;       LXI     D,FCB
                        ;       CALL    CF
                        ;       JC      ERROR

0013 =          B$ERASE         EQU     19      ;Borrar fichero
0016 =          B$CREATE        EQU     22      ;Crear fichero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                CF:
0000 D5                 PUSH    D               ;Conservación apuntador al FCB
0001 0E13               MVI     C,B$ERASE       ;Borrado de cualquier otro fichero
0003 CD0500             CALL    BDOS
0006 D1                 POP     D               ;Recuperación apuntador a FCB
0007 0E16               MVI     C,B$CREATE      ;Creación (y abrir) nuevo fichero
0009 CD0500             CALL    BDOS
000C FEFF               CPI     OFFH            ;Acarreo a uno si OK, a 0 si error
000E 3F                 CMC                     ;Completar usando el acarreo a 1 si error
000F C9                 RET
```

Figura 5-20. Petición de creación de fichero.

No deben usarse nunca nombres de fichero ambiguos en una operación de cambio de nombre; produce efectos extraños y puede resultar que los ficheros queden dañados irremediablemente. Esta función cambiará *todas* las versiones del antiguo nombre al nombre nuevo.

La figura 5-21 muestra una subrutina que aceptará un nombre y tipo de fichero existente y un nuevo nombre y tipo volverá a denominar el fichero antiguo de acuerdo con el nuevo nombre. Controla que el nuevo nombre de fichero no exista ya, devolviendo un código de error si es así.

```
;RF
;Cambio de nombre de fichero
;Esta subrutina cambia de nombre un fichero.
;Usa la rutina BF (construcción FCB) mostrada en Fig. 5.16

;Parámetros de entrada
;       *** No hay cambio de minúsculas a mayúsculas ***
;       HL -> nombre de fichero a cambiar (terminado por 00H)
;       DE -> nuevo nombre (terminado por 00H)

;Parámetros de salida
;       Acarreo a cero si éxito en operación
;                       (A = 0,1,2,3)
;       Acarreo a uno si error
;               A = 0FEH si existe ya el nuevo fichero
;               A = 0FFH si no existe el fichero a cambiar
```

Figura 5-21. Petición de cambio de nombre de fichero.

```
                ;Secuencia de llamada
                ;       LXI     H,OLDNAME       ;HL -> nombre antiguo
                ;       LXI     D,NEWNAME       ;DE -> nombre nuevo
                ;       CALL    RF
                ;       JC      ERROR

000F =          B$OPEN          EQU     15      ;Abrir fichero
0017 =          B$RENAME        EQU     23      ;Cambio de nombre de fichero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 00000000000RFFCB:  DW      0,0,0,0,0,0,0,0 ;1 1/2 FCB's long
0010 0000000000         DW      0,0,0,0,0,0,0,0
0020 0000000000         DW      0,0,0,0,0,0,0,0
0030 000000             DW      0,0,0

                RF:
0036 D5                 PUSH    D               ;Guardar apuntador a nuevo nombre
0037 110000             LXI     D,RFFCB         ;Construcción FCB nombre antiguo
                                                ;HL -> nombre antiguo
003A CD5D00             CALL    BF

003D E1                 POP     H               ;Recuperación apuntador nuevo nombre
003E 111000             LXI     D,RFFCB+16      ;Componer nuevo nombre en la segunda parte
0041 CD5D00             CALL    BF              ;del bloque de control de fichero

0044 111000             LXI     D,RFFCB+16      ;Intento experimental
0047 0E0F               MVI     C,B$OPEN        ;de abrir el nuevo fichero
0049 CD0500             CALL    BDOS            ;para asegurar
004C FEFF               CPI     0FFH            ;que no existe ya
004E 3EFE               MVI     A,0FEH          ;Suposición de error (marcas no cambiadas)
0050 D8                 RC                      ;Acarreo a uno si A es 0,1,2,3

0051 110000             LXI     D,RFFCB         ;Cambio de nombre del fichero
0054 0E17               MVI     C,B$RENAME
0056 CD0500             CALL    BDOS
0059 FEFF               CPI     0FFH            ;Acarreo a 1 si OK, a 0 si error
005B 3F                 CMC                     ;Inversión uso del acarreo, a 1 si error
005C C9                 RET

                ;BF
                ;Construcción bloque de control de fichero
                ;Esta subrutina compone una cadena terminada
                ;por un byte 00H en el FCB, poniendo disco
                ;y nombre y tipo de fichero y poniendo a cero
                ;el resto del FCB

                ;Parámetros de entrada
                ;       DE -> bloque de control de fichero (36 bytes)
                ;       HL -> cadena nombre de fichero (terminada con un byte 00H)

                ;Parámetros de salida
                ;       Bloque de control de fichero construido

                ;Secuencia de llamada
                ;       LXI     D,FCB
                ;       LXI     H,FILENAME
                ;       CALL    BF

                BF:
005D C9                 RET                     ;Subrutina ficticia: Ver Figura 5-16
```

Figura 5-21. (Continuación.)

Función 24: Obtención de discos activos (vector de apertura)

Código de función: C=18H
Parámetros de entrada: Ninguno
Parámetros de salida: HL=Activación del mapa de disco (vector de apertura)

Ejemplo

```
0018 =          B$GETACTDSK     EQU     24      ;Obtención discos activos
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                                                ;Ejemplo de obtención del código
0000 0E18               MVI     C,B$GETACTDSK   ; de función de disco activo
0002 CD0500             CALL    BDOS            ;HL = mapa de bits disco activo
                                                ;Los bits son = 1 si disco activo
                                                ;Bits 15 14 13 ... 2 1 0
                                                ;Disco P  O  N ... C B A
```

Propósito Esta función devuelve un mapa de bits, denominado *vector de apertura*, al par de registros HL, indicando qué controlador de disco lógico ha sido seleccionado desde el último arranque caliente o desde la última llamada a la función de puesta a cero del disco (código 13, 0DH). El bit menos significativo de L corresponde al disco A, mientras que el bit de mayor orden de H corresponde al disco P. El bit correspondiente al disco lógico específico está puesto a 1 si el disco ha sido seleccionado o a 0 si el disco no está actualmente en línea.

Los discos lógicos pueden ser seleccionados por programa a través de una operación de ficheros que pone el campo de unidad en un valor distinto de cero, a través de la función de selección de disco (código 14, 0EH), o por el operador por introducción de una orden "X:", donde "X" es igual a A, B, ..., P.

Notas Esta función está pensada para los programas que necesitan saber qué discos lógicos están activos actualmente en el sistema —es decir, aquellos discos lógicos que han sido seleccionados.

Función 25: Obtención del disco actual por defecto

Código de función: C=19H
Parámetros de entrada: Ninguno
Parámetros de salida: A=Disco actual
(0=A, 1=B, ..., F=P)

Ejemplo

```
0019 =          B$GETCURDSK     EQU     25      ;Obtención disco en curso
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E19               MVI     C,B$GETCURDSK   ;Código de función
0002 CD0500             CALL    BDOS            ;A = 0 si A:, 1 si B: ...
```

Propósito Esta función devuelve el disco actual por defecto fijado por medio de la última llamada a la función de selección de disco (código 14, 0EH) o por la introducción al CCP de una orden "X:" por el operador, donde "X" es A, B, ..., P.

Notas Esta función devuelve el disco actual por defecto en forma codificada. Registro A=0 si la unidad A es la actual, 1 si es B, etc. Si se necesita convertir esto a los caracteres ASCII correspondientes, hay que añadir simplemente 41H al registro A.

Utilícese esta función para convertir el nombre y tipo de fichero de un FCB a una cadena ASCII para visualizarla. Si el primer byte del FCB es 00H, se está utilizando el controlador por defecto. Por consiguiente, habrá de utilizarse esta función para determinar la letra lógica del disco para el disco por defecto.

Función 26: Fijación de la dirección DMA (lectura/escritura)

Código de función: C=1AH
Parámetros de entrada: DE=Dirección de DMA (lectura/escritura)
Parámetros de salida: Ninguno

Ejemplo

```
001A =          B$SETDMA        EQU     26      ;Fijar dirección DMA
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS
0000            SECBUFF:        DS      128     ;Memoria intermedia de sector

0080 0E1A               MVI     C,B$SETDMA      ;Código de función
0082 110000             LXI     D,SECBUFF       ;Apuntador a memoria intermedia
0085 CD0500             CALL    BDOS
```

Propósito Esta función coloca la dirección de acceso directo a memoria del BDOS (DMA) en un nuevo valor. El nombre es una reliquia histórica que data de los tiempos del sistema de desarrollo Intel, en el que se desarrolló originalmente CP/M. Esta máquina, en virtud de su hardware, podía leer datos de un disquete directamente en la memoria o escribir datos en un disquete directamente desde la memoria. El nombre de *dirección DMA* se aplica ahora a la dirección de la memoria intermedia a la cual y desde la cual se transfieren los datos siempre que un disquete lee, escribe o cuando se realiza una operación de directorio.

Siempre que el CP/M se pone en marcha por primera vez (arranque en frío) o se realiza arranque caliente o se llevan a cabo operaciones de puesta a cero del disco, la dirección del DMA se pone en su valor por defecto 0080H.

Notas Ninguna llamada de función puede informar acerca del valor actual de la dirección DMA. Todo lo que puede hacerse es una llamada a la función que fija la dirección DMA para asegurar que esté cuando se necesite.

Una vez que se ha fijado la dirección DMA en el lugar correcto para el programa, quedará fijada allí hasta que se realice otra llamada a la función, puesta a cero del disco o arranque caliente.

Las operaciones de lectura y escritura secuencial y aleatoria utilizan el valor en curso de la dirección del DMA como lo hacen las operaciones de directorio búsqueda del primer caso y búsqueda del próximo.

Función 27: Obtención del vector de situación

Código de función: C=1BH
Parámetros de entrada: Ninguno
Parámetros de salida: HL=Dirección del vector de situación

Ejemplo

```
001B =          B$GETALVEC      EQU     27      ;Obtención dirección de vector de situación
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E1B               MVI     C,B$GETALVEC    ;Código de función
0002 CD0500             CALL    BDOS            ;HL -> dirección base del
                                                ;      vector de situación
```

Propósito Esta función devuelve la dirección base, o de comienzo, del vector de situación para el disco lógico actualmente seleccionado. Esta información, que indica las partes del disco que están asignadas, se utiliza por los programas de utilidad y por el propio BDOS para determinar cuánto espacio no utilizado existe en el disco lógico, para localizar un bloque no utilizado para extender un fichero, o para abandonar un bloque cuando se borra un fichero.

Notas Digital Research considera la disposición actual del vector de situación como información propia.

Función 28: Situación del disco lógico en modo de sólo lectura

Código de función: C=1CH
Parámetros de entrada: Ninguno
Parámetros de salida: Ninguno

Ejemplo

```
001C =          B$SETDSKRO      EQU     28      ;Puesta del disco en sólo lectura
                                                ; código de función
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                                                ;Fijación de los discos seleccionados
                                                ; anteriormente
                                                ;Selección de llamada de función
                                                ; de disco
0000 0E1C               MVI     C.B$SETDSKRO    ;Código de función
0002 CD0500             CALL    BDOS
```

Propósito Esta función fija lógicamente el disco seleccionado a un estado de sólo lectura. Cualquier intento de realizar una función de escritura secuencial o al azar en el disco seleccionado será interceptada por el BDOS y aparecerá en la consola el mensaje siguiente:

```
BDOS Err on X: R/O
```

donde X: es el disco seleccionado.

Notas Una vez se ha requerido el modo de sólo lectura para el disco lógico actualmente seleccionado, ésta permanecerá aun en caso de que se proceda a seleccionar otros discos lógicos. Efectivamente, estará en vigor hasta el próximo arranque caliente o hasta la llamada a la función de puesta a cero del sistema de disco.

La documentación de Digital Research se refiere a este código de función como a protección de escritura del disco. La descripción de sólo lectura se utiliza aquí porque corresponde al mensaje de error producido si el programa intenta escribir en el disco.

Función 29: Obtención de los discos en estado de sólo lectura

Código de función: C=1DH
Parámetros de entrada: Ninguno
Parámetros de salida: HL=Mapa de disco de sólo lectura

Ejemplo

```
001D =          B$GETRODSKS     EQU     29      ;Obtención de discos en modo sólo lec.
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

0000 0E19               MVI     C,B$GETRODSKS   ;Código de función
0002 CD0500             CALL    BDOS            ;HL = mapa de bits de discos en sólo
                                                ; lectura
                                                ;Los bits están a 1 si disco en sólo
                                                ; lectura
                                                ;Bits 15 14 13 ... 2 1 0
                                                ;Disco P  O  N ... C B A
```

Propósito Esta función devuelve un mapa de bits en los registros H y L mostrando qué discos lógicos del sistema se han puesto en modo de sólo lectura, bien por una llamada a la función de situación del disco en modo de sólo lectura (código 28, 1CH), o por el propio BDOS, porque ha detectado que se ha cambiado un disquete.

El bit menos significativo de L corresponde al disco lógico A, mientras que el bit más significativo de H corresponde al disco P. El bit que corresponde al disco lógico específico está puesto a 1 si el disco se ha puesto en modo de sólo lectura.

Función 30: Fijación de atributos de fichero

Código de función: C=1EH
Parámetros de entrada: DE=Dirección del FCB
Parámetros de salida: A=Código de directorio

Ejemplo

```
001E =          B$SETFAT        EQU     30      ;Fijar atributo de fichero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda en la unidad de disco por defecto
0001 46494C454EFCB$NAME:        DB      'FILENAME'      ;Nombre de fichero
0009 D4         FCB$TYP:        DB      'T'+80H         ;Tipo con
                                                        ; atributo R/O
000A 5950                       DB      'YP'
000C 0000000000                 DW      0,0,0,0,0,0,0,0,0,0,0

0022 0E1E               MVI     C,B$SETFAT      ;Código de función
0024 110000             LXI     D,FCB           ;DE -> bloque de control de fichero
                                                ;Bits MS a uno en nombre/tipo de fichero
0027 CD0500             CALL    BDOS            ;A = 0FFH si no fichero no encontrado
```

Propósito Esta función fija los bits que describen los atributos de un fichero en las entradas relevantes del directorio para un fichero específico. Cada fichero puede contener hasta 11 atributos. De estos 11, dos tienen significados predefinidos, otros cuatro están disponibles para ser utilizados por el programador, y los cinco restantes están reservados para usos futuros del CP/M.

Cada atributo consiste en un bit único. El bit más significativo de cada byte del nombre y tipo de fichero se utiliza para almacenar los atributos. Los atributos de fichero se designan por un código consistente en la letra "f" (para nombre de fichero) o "t" (para tipo), seguida del número de posición del carácter y una comilla única. Por ejemplo, el atributo de sólo lectura es t1'.

El significado de los atributos es el siguiente:

- f1' a f4' Disponible para ser utilizado por el programador
- f5' a f8' Reservado para uso futuro del CP/M
- t1' Atributo de sólo lectura del fichero
- t2' Atributo de fichero modo sistema
- t3' Reservado para uso futuro del CP/M

Los atributos están fijados presentando esta función con un FCB en el cual el nombre no ambiguo del fichero ha sido fijado previamente con los bits más significativos colocados apropiadamente. Esta función busca entonces el directorio para una comparación y cambia las entradas comparadas para que contengan los atributos que se han fijado en el FCB.

El BDOS interceptará cualquier intento de escribir en un fichero que tiene puesto el atributo de sólo lectura. La orden DIR del CCP no muestra ningún fichero en modo sistema.

Notas Se pueden usar los cuatro atributos disponibles para poner en marcha un sistema de seguridad de ficheros o quizá para marcar ciertos ficheros que deben mantener otros discos. Las funciones de búsqueda del primer caso y búsqueda del siguiente permiten visualizar la entrada completa del directorio, de forma que los programas pueden comprobar fácilmente los atributos.

Los ejemplos de subrutinas de las figuras 5-22 y 5-23 muestran cómo fijar los atributos de fichero (SFA) y obtener atributos de fichero (GFA), respectivamente. Ambos utilizan un mapa de bits en el cual los 11 bits más significativos del par de registros HL se utilizan para indicar los bits correspondientes de los 11 caracteres de la combinación nombre/tipo de fichero. Se verán entonces algunas equivalencias que hacen más fácil la manipulación de los atributos en este mapa de bits.

```
                ;SFA
                ;Fijación de atributos de fichero
                ;Esta subrutina toma un mapa de bits comprimido de todos los bits
                ;de atributos del fichero, lo expande en un bloque de control
                ;de fichero existente y hace una petición a CP/M para activar
                ;los atributos en el directorio de ficheros.

                ;Parámetros de entrada
                ;       DE -> bloque de control de fichero
                ;       HL -> mapa de bits. Sólo se utilizan los bits más
                ;             significativos que corresponden directamente
                ;             a los posibles bytes de atributos

                ;Parámetros de salida
                ;       Acarreo a cero si éxito en la operación (A = 0,1,2,3)
                ;       Acarreo a uno si error (A = 0FFH)

                ;Secuencia de llamada
                ;       LXI     D,FCB
                ;       LXI     H,0000$0000$1100$0000B  ;Mapa de bits
                ;       CALL    SFA
                ;       JC      ERROR

                                                ;Equivalencias de atributos de fichero

8000 =          FA$F1   EQU     1000$0000$0000$0000B    ;F1' - F4'
4000 =          FA$F2   EQU     0100$0000$0000$0000B    ;Disponible para su uso por
2000 =          FA$F3   EQU     0010$0000$0000$0000B    ; programas de aplicación
1000 =          FA$F4   EQU     0001$0000$0000$0000B

0800 =          FA$F5   EQU     0000$1000$0000$0000B    ;F5' - F8'
0400 =          FA$F6   EQU     0000$0100$0000$0000B    ;Reservado para CP/M
0200 =          FA$F7   EQU     0000$0010$0000$0000B
0100 =          FA$F8   EQU     0000$0001$0000$0000B

0080 =          FA$T1   EQU     0000$0000$1000$0000B    ;T1' -- fichero sólo lectura
0080 =          FA$RO   EQU     FA$T1
0040 =          FA$T2   EQU     0000$0000$0100$0000B    ;T2' -- fichero modo sistema
0040 =          FA$SYS  EQU     FA$T2
0020 =          FA$T3   EQU     0000$0000$0010$0000B    ;T3' -- reservado para CP/M

001E =          B$SETFAT        EQU     30      ;Fijar atributos de fichero
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS
```

Figura 5-22. Fijación de atributos de fichero.

```
                SFA:
0000 D5                 PUSH    D               ;Guardar apuntador a FCB
0001 13                 INX     D               ;HL -> primer carácter nombre de fichero
0002 0E0B               MVI     C,8+3           ;Bucle de conteo para nombre y tipo de fichero

                SFAL:                           ;Bucle principal de proceso
0004 AF                 XRA     A               ;Borrado de A y acarreo
0005 29                 DAD     H               ;Desplazamiento del siguiente bit MS en el acarreo
0006 CE00               ACI     0               ;A = 0,1 dependiendo del acarreo
0008 0F                 RRC                     ;Rotación del bit LS de A en bit NS
0009 47                 MOV     B,A             ;Guardar resultado (00H o 80H)
000A EB                 XCHG                    ;HL -> carácter en FCB
000B 7E                 MOV     A,M             ;Obtención de carácter en FCB
000C E67F               ANI     7FH             ;Aislar todo excepto bit de atributo
000E B0                 ORA     B               ;Puesta a uno del atributo en base al resultado
000F 77                 MOV     M,A             ;y almacenar en FCB
0010 EB                 XCHG                    ;DE -> FCB, HL = resto de mapa de bits
0011 13                 INX     D               ;DE -> carácter siguiente en el FCB
0012 0D                 DCR     C               ;Descuento de carácter en contador
0013 C20400             JNZ     SFAL            ;Salto atrás para nuevo carácter
0016 0E1E               MVI     C,B$SETFAT      ;Activar código de función de atributo en fichero
0018 D1                 POP     D               ;Recuperar apuntador a FCB
0019 CD0500             CALL    BDOS
001C FEFF               CPI     0FFH            ;Acarreo a uno si OK, a cero si error
001E 3F                 CMC                     ;Inversión para utilizar acarreo a 1 si error
001F C9                 RET
```

Figura 5-22. (Continuación.)

```
                ;GFA
                ;Obtención de atributos de fichero
                ;Esta subrutina busca el fichero deseado utilizando la rutina
                ;de búsqueda del primero antes de abrir el fichero. Construye
                ;entonces un mapa de bits con los bits de atributos en el nombre
                ;y tipo de fichero. Con este mapa y el mapa de bits de entrada
                ;se hará la operación AND y devolverá el resultado en el indicador
                ;de cero. El mapa real será también devuelto en caso
                ;de que se necesite una comprobación más compleja.

                ;>>>    Nota: Esta subrutina cambia la dirección DMA de CP/M.

                ;Parámetros de entrada
                ;       DE -> bloque de control de fichero
                ;       HL = máscara de mapa de bits para ser operado (AND)
                ;            con los atributos

                ;Parámetros de salida
                ;       Acarreo a cero. Operación terminada con éxito
                ;               Estado distinto de cero como resultado de la operación AND
                ;               entre la máscara y el conjunto de bits de atributo.
                ;               HL = conjunto de bytes de atributo no enmascarados.
                ;       Acarreo a uno, fichero no encontrado

001A =          B$SETDMA        EQU     26      ;Fijación dirección DMA
0011 =          B$SEARCHF       EQU     17      ;Búsqueda de la primera entrada para coincidencia
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS
0080 =          GFADMA          EQU     80H     ;Dirección DMA por defecto

                ;Secuencia de llamada
                ;       LXI     D,FCB
                ;       LXI     H,0000$0000$1100$0000B  ;Mapa de bits
                ;       CALL    GFA
                ;       JC      ERROR

                                                ;Equivalencias atributos de fichero

8000 =          FA$F1   EQU     1000$0000$0000$0000B    ;F1' - F5'
4000 =          FA$F2   EQU     0100$0000$0000$0000B    ;Disponible para su uso por
```

Figura 5-23. Obtención de atributos de fichero.

```
2000 =          FA$F3   EQU     0010$0000$0000$0000B    ;Programas de aplicación
1000 =          FA$F4   EQU     0001$0000$0000$0000B

0800 =          FA$F5   EQU     0000$1000$0000$0000B    ;F6' - F8'
0400 =          FA$F6   EQU     0000$0100$0000$0000B    ;Reservado para CP/M
0200 =          FA$F7   EQU     0000$0010$0000$0000B
0100 =          FA$F8   EQU     0000$0001$0000$0000B

0080 =          FA$T1   EQU     0000$0000$1000$0000B    ;T1' --  fichero de sólo lectura
0080 =          FA$RO   EQU     FA$T1
0040 =          FA$T2   EQU     0000$0000$0100$0000B    ;T2' --  fichero en modo sistema
0040 =          FA$SYS  EQU     FA$T2
0020 =          FA$T3   EQU     0000$0000$0010$0000B    ;T3' --  reservado para CP/M

                GFA:
0000 E5                 PUSH    H               ;Almacenamiento máscara-AND
0001 D5                 PUSH    D               ;Almacenamiento apuntador a FCB
0002 0E1A               MVI     C,B$SETDMA      ;Puesta de DMA a dirección por defecto
0004 118000             LXI     D,GFADMA        ;DE -> dirección DMA
0007 CD0500             CALL    BDOS

000A D1                 POP     D               ;Recuperación apuntador a FCB
000B 0E11               MVI     C,B$SEARCHF     ;Búsqueda de coincidencia con nombre
000D CD0500             CALL    BDOS
0010 FEFF               CPI     OFFH            ;Acarreo a 1 si OK, a cero si error
0012 3F                 CMC                     ;Inversión para usar acarreo a 1 si error
0013 DA4100             JC      GFAX            ;Retorno si error
                                                ;Multiplicar por 32 para obtener el desplazamiento
                                                ; en la memoria intermedia DMA
0016 87                 ADD     A               ;* 2
0017 87                 ADD     A               ;* 4
0018 87                 ADD     A               ;* 8
0019 87                 ADD     A               ;* 16
001A 87                 ADD     A               ;* 32
001B 5F                 MOV     E,A             ;Composición del valor sobre una palabra
001C 1600               MVI     D,0             ;HL -> dirección DMA
001E 218000             LXI     H,GFADMA        ;HL -> entrada al directorio en mem. int. DMA
0021 19                 DAD     D               ;HL -> primer carácter de nombre de fichero
0022 23                 INX     H               ;DE -> primer carácter de nombre de fichero
0023 EB                 XCHG

0024 0E0B               MVI     C,8+3           ;Cuenta de caracteres en nombre y tipo de fichero
0026 210000             LXI     H,0             ;Borrado del mapa de bits

                GFAL:                           ;Bucle principal
0029 1A                 LDAX    D               ;Búsqueda del carácter siguiente de nombre de fichero
002A E680               ANI     80H             ;Aislamiento del bit de atributo
002C 07                 RLC                     ;Transferencia del bit MS en el bit LS
002D B5                 ORA     L               ;OR del conjunto de bits previo
002E 6F                 MOV     L,A             ;Guardar resultado
002F 29                 DAD     H               ;Desplazamiento de HL 1 bit a la izquierda cada vez
0030 13                 INX     D               ;DE -> carácter siguiente en nombre tipo de fichero
0031 0D                 DCR     C               ;Descontar del contador
0032 C22900             JNZ     GFAL            ;Vuelta atrás para carácter siguiente

0035 29                 DAD     H               ;Justificación a la izquierda de los bits de atributo en HL
0036 29                 DAD     H               ;El bit de atributo MS estará ya
0037 29                 DAD     H               ;en el onceavo bit de HL, hacen falta, por tanto, sólo
0038 29                 DAD     H               ;cuatro desplazamientos

0039 D1                 POP     D               ;Recuperación máscara-AND
003A 7A                 MOV     A,D             ;Obtención byte MS de resultado
003B A4                 ANA     H               ;AND con el byte MS de resultado
003C 47                 MOV     B,A             ;Guardar resultado intermedio
003D 7B                 MOV     A,E             ;Obtención byte LS de la máscara
003E A5                 ANA     L               ;AND con el byte LS de resultado
003F B0                 ORA     B               ;Combinar ambos resultados para fijar el indicador Z

0040 C9                 RET

                GFAX:                           ;Salida de error
0041 E1                 POP     H               ;Pop de la pila
0042 C9                 RET
```

Figura 5-23. (Continuación.)

Función 31: Obtención de la dirección del bloque de parámetros de disco

Código de función: C=1FH
Parámetros de entrada: Ninguno
Parámetros de salida: HL=Dirección del DPB

Ejemplo

```
001F =          B$GETDPB        EQU     31      ;Obtención de la dirección del bloque
                                                ; de parámetros del disco
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                                                ;Devuelve la dirección DPB del disco
                                                ; seleccionado previamente
                                                ; con la función selección
                                                ; de disco
0000 0E1F               MVI     C,B$GETDPB      ;Código de función
0002 CD0500             CALL    BDOS            ;HL -> dirección de comienzo del bloque
                                                ; de parámetros del disco en curso
```

Propósito Esta función devuelve la dirección del bloque de parámetros de disco (DPB) para el último disco lógico seleccionado. El DPB, explicado en el capítulo 3, describe las características de un disco lógico específico —información de interés principalmente para programas de utilidad del sistema.

Notas Las subrutinas presentadas en la figura 5-24 se enfrentan con dos graves problemas. Primero, dado un número de pista y de sector, ¿en qué bloque se introducirá? Y a la inversa, dado un bloque, ¿cuáles son su pista y su sector iniciales?

Estas subrutinas se utilizan normalmente por las utilidades del sistema. Primero obtienen la dirección del DPB usando esta función del BDOS. Entonces conmutan para utilizar llamadas directas del BIOS para realizar sus otras funciones, tales como selección de discos, pistas y sectores y para leer y escribir en el disco.

La primera subrutina, GTAS (obtención de pista y sector), en la figura 5-25, toma un número de bloque y lo convierte para proporcionar los números de pista y sector iniciales. El GMTAS (obtención de pista y sector máximos) devuelve el número máximo de pista y sector para el disco especificado. El GDTAS (obtención de pistas y sectores del directorio) indica no sólo la pista y el sector iniciales del directorio, sino también el número de sectores de 128 bytes del directorio.

Obsérvese que siempre que un número de pista se utiliza como parámetro de entrada o salida, es un número de pista absoluto. Es decir, el número de pistas reservadas del disco antes del directorio ha sido agregado a él.

GNTAS (obtención de la pista y el sector siguientes) ayuda a leer sectores secuencialmente. Añade 1 al número de sector, y cuando se alcanza el final de una pista, actualiza el número de pista añadiendo 1 y pone a 1 el número del sector.

```
                ;Subrutina de utilidad para acceder a datos situados
                ;en el bloque de parámetros del disco

000E =          B$SELDSK        EQU     14      ;Selección del código de función del disco
001F =          B$GETDPB        EQU     31      ;Obtención dirección DPB
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                ;Hace un código más compacto, por razones de facilidad,
                ;para copiar el bloque de parámetros del disco específico
                ;en variables locales mientras se manipula la información.
                ;Estas variables son las siguientes:

                DPB:                                    ;Bloque de parámetros del disco
0000 0000       DPBSPT: DW      0                       ;Sectores de 128 bytes por pista
0002 00         DPBBS:  DB      0                       ;Desplazamiento de bloques
0003 00         DPBBM:  DB      0                       ;Máscara de bloques
0004 00         DPBEM:  DB      0                       ;Máscara de extensión
0005 0000       DPBMAB: DW      0                       ;Número máximo de bloque
0007 0000       DPBNOD: DW      0                       ;Número de entradas al directorio-1
0009 0000       DPBDAB: DW      0                       ;Bloques del directorio
000B 0000       DPBCBS: DW      0                       ;Comprobación tamaño memoria intermedia
000D 0000       DPBTBD: DW      0                       ;Pistas reservadas antes del directorio

000F =          DPBSZ   EQU     $-DPB                   ;Tamaño del bloque de parámetros del disco

                ;GETDPB
                ;Obtención bloque de parámetros del disco
                ;Esta subrutina copia el DPB del disco lógico especificado
                ;en las variables locales para DPB mencionadas.

                ;Parámetros de entrada
                ;       A = número de disco lógico (A: = 0, B: = 1...)

                ;Parámetros de salida
                ;       Variables locales conteniendo DPB

                GETDPB:
000F 5F                 MOV     E,A             ;Obtención del código de disco del disco seleccionado
0010 0E0E               MVI     C,B$SELDSK      ;Selección del disco
0012 CD0500             CALL    BDOS
0015 0E1F               MVI     C,B$GETDPB      ;Obtención de la dirección de comienzo de los parámetros
                                                ;de disco
0017 CD0500             CALL    BDOS            ;HL -> DPB
001A 0E0F               MVI     C,DPBSZ         ;Activación del contador
001C 110000             LXI     D,DPB           ;Obtención de la dirección de comienzo de las variables
                                                ;locales
                GDPBL:                          ;Copia del DPB en las variables locales
001F 7E                 MOV     A,M             ;Obtención de un byte del DPB
0020 12                 STAX    D               ;Almacenar en una variable local
0021 13                 INX     D               ;Actualización apuntador a variable local
0022 23                 INX     H               ;Actualización apuntador DPB
0023 0D                 DCR     C               ;Descontar del contador
0024 C21F00             JNZ     GDPBL           ;Vuelta atrás para byte siguiente
0027 C9                 RET

                ;GTAS
                ;Obtención de pista y sector (dado un número de bloque)

                ;Esta subrutina convierte un número de bloque
                ;en un número de pista y sector -- nótese que se basa en
                ;sectores de 128 bytes

                ;>>>>>   Nota: Debe llamarse a GETDPB antes de
                ;>>>>>         llamar a esta subrutina

                ;Parámetros de entrada
                ;       HL = número de bloque

                ;Parámetros de salida
                ;       HL = número de pista
                ;       DE = número de sector
```

Figura 5-24. Acceso a los datos del bloque de parámetros del disco.

```
                ;Método:
                ;En términos matemáticos la pista puede ser deducida:
                ;Trk = (Bloque * sec. por bloque) / sec. por pista
                ;       + pistas antes del directorio
                ;El sector se deduce en la forma:
                ;Sec = ((Bloque * sec. por bloque) módulo/
                ;       sec. por pista) + 1

                GTAS:
0028 3A0200             LDA     DPBBS           ;Obtención bloque desplazado -- de 3 a 7
                                                ;dependiendo del tamaño de bloque
                                                ;Será usado como contador para desplazamiento

                GTASS:
002B 29                 DAD     H               ;Desplazamiento del bloque un lugar a la izquierda
002C 3D                 DCR     A               ;Decremento del contador de desplazamiento de bloque
002D C22B00             JNZ     GTASS           ;Necesidad de más desplazamientos
0030 EB                 XCHG                    ;DE = bloque * sec. por bloque
                                                ;por ejemplo, DE = número total de sectores
0031 2A0000             LHLD    DPBSPT          ;Obtención de sectores/pista
0034 EB                 XCHG                    ;HL = sec. por pista, DE = número total de sectores
0035 CD8F00             CALL    DIVHL           ;BC = HL/DE, HL = resto
                                                ;BC = pista, HL = sector
0038 23                 INX     H               ;La numeración de sectores empieza por 1
0039 EB                 XCHG                    ;DE = sector, HL = pista
003A 2A0D00             LHLD    DPBTBD          ;Pistas antes del directorio
003D 09                 DAD     B               ;DE = sector, HL = pista absoluta
003E C9                 RET

                ;GMTAS
                ;Obtención de pista y sector máximos

                ;Es únicamente una llamada a GTAS con el bloque
                ;máximo como parámetro de entrada

                ;>>>>>   Nota: Debe llamarse a GETDPB antes
                ;>>>>>         de llamar a esta subrutina

                ;Parámetros de entrada: Ninguno

                ;Parámetros de salida
                ;       HL = número máximo de pista
                ;       DE = sector máximo
                GMTAS:
003F 2A0500             LHLD    DPBMAB          ;Obtención bloque máximo
0042 C32800             JMP     GTAS            ;Retorno a GTAS con los parámetros en HL y DE

                ;GDTAS
                ;Obtención de pista y sector del directorio

                ;Devuelve la pista y sector de comienzo
                ;del directorio de ficheros junto con el número
                ;de sectores del directorio.

                ;>>>>>  Nota: Debe llamarse a GETDPB antes
                ;>>>>>        de llamar a esta subrutina

                ;Parámetros de entrada: Ninguno

                ;Parámetros de salida
                ;       BC = número de sectores del directorio
                ;       DE = sector de comienzo del directorio
                ;       HL = pista de comienzo del directorio

                GDTAS:
0045 2A0700             LHLD    DPBNOD          ;Obtención de número de entrada al direc.-1
0048 23                 INX     H               ;Componer número verdadero de entradas
                                                ;Cada entrada tiene 32 bytes de longitud, dividir
                                                ;por cuatro para convertir a sec. de 128 bytes
0049 CDD000             CALL    SHLR            ;/ 2 (desplazando HL a la derecha 1 bit)
004C CDD000             CALL    SHLR            ;/ 4
004F E5                 PUSH    H               ;Guardar número de sectores
0050 210000             LXI     H,0             ;El directorio empieza en bloque 0
0053 CD2800             CALL    GTAS            ;HL = pista, DE = sector
0056 C1                 POP     B               ;Recuperar número de sectores
0057 C9                 RET
```

Figura 5-24. (Continuación.)

```
                ;GNTAS
                ;Obtención pista y sector de siguiente

                ;Esta subrutina actualiza la pista y sector de entrada
                ;añadiendo una unidad, incrementando el número de pista
                ;y situando el número de sector en la forma adecuada.

                ;>>>>>  Nota: Debe llamarse a GETDPB antes
                ;>>>>>        de llamar a esta subrutina

                ;Nota: Debe comprobarse final-de-disco comparando
                ;      el número de pista devuelto por esta rutina
                ;      con el devuelto por GMTAS + 1. Si coinciden
                ;      se ha llegado al final del disco.

                ;Parámetros de entrada
                ;        HL = número de pista en curso
                ;        DE = número de sector en curso

                ;Parámetros de salida
                ;        HL = número de pista actualizado
                ;        DE = número de sector actualizado

                GNTAS:

0058 E5                 PUSH    H               ;Guardar pista
0059 13                 INX     D               ;Actualizar sector
005A 2A0000             LHLD    DPBSPT          ;Obtener sectores por pista
005D CDC900             CALL    SUBHL           ;HL = HL - DE
0060 E1                 POP     H               ;Recuperar pista en curso
0061 D0                 RNC                     ;Retorno si sector act. <= sec. por pista
0062 23                 INX     H               ;Actualizar pista sec. act. > sec. por pista
0063 110100             LXI     D,1             ;Puesta a 1 del sector
0066 C9                 RET

                ;GAB
                ;Obtención de bloque

                ;Esta rutina devuelve un número de bloque
                ;dado un número de pista y de sector. Da también
                ;el desplazamiento del bloque en el que puede
                ;encontrarse el sector. Este desplazamiento viene dado
                ;en unidades de sectorres de 128 bytes.

                ;>>>>>  Nota: Debe llamarse a GETDPB antes
                ;>>>>>        de llamar a esta subrutina

                ;Parámetros de entrada
                ;        HL = número de pista
                ;        DE = número de sector

                ;Parámetros de salida
                ;        HL = número de bloque

                ;Método
                ;El número de bloque se obtiene:
                ;AB = (sector + ((pista - pistas antes del directorio)
                ;        * sectores por pista)) / log 2 (sectores por bloque)

                ;El desplazamiento de sector de un bloque se calcula de la forma siguiente:
                ;Despl. = (sector * ((pista - pistas antes del directorio)
                ;        * sectores por pista)) / AND (sectores por bloque-1)

                GAB:
0067 D5                 PUSH    D               ;Guardar sector
0068 EB                 XCHG                    ;DE = pista
0069 2A0D00             LHLD    DPBTBD          ;Obtención número de pistas antes del directorio
006C EB                 XCHG                    ;DE = número de pista antes dir. HL = pista
006D CDC900             CALL    SUBHL           ;HL = HL - DE
                                                ;HL = pista relativa en disco lógico
0070 EB                 XCHG                    ;DE = pista relativa
0071 2A0000             LHLD    DPBSPT          ;Obtención sectores por pista
0074 CDA400             CALL    MULHL           ;HL = HL * DE
                                                ;HL = número de sectores
0077 EB                 XCHG                    ;DE = número de sectores
```

Figura 5-24. (Continuación.)

```
0078 E1              POP     H          ;Recuperar sector
0079 2B              DCX     H          ;Hacerlo relativo a 0
007A 19              DAD     D          ;HL = sector relativo
007B 3A0300          LDA     DPBBM      ;Obtención máscara de bloque
007E 47              MOV     B,A        ;Preparado para operación AND
007F 7D              MOV     A,L        ;Obtención byte LS de sector relativo
0080 A0              ANA     B          ;AND con máscara de bloques
0081 F5              PUSH    PSW        ;A = desplazamiento de sector
0082 3A0200          LDA     DPBBS      ;Obtención desplazamiento bloque
0085 4F              MOV     C,A        ;Puesta en contador

                GABS:                   ;Bucle de desplazamiento
0086 CDD000          CALL    SHLR       ;HL desplazado a la derecha (dividido por 2)
0089 0D              DCR     C          ;Descontar
008A C28600          JNZ     GABS       ;Desplazar más si es necesario
008D F1              POP     PSW        ;Recuperación del desplazamiento en dirección
008E C9              RET

                ;Subrutinas de utilidad
                ;Esta subrutina realiza la aritmética de 16 bits sobre el par de registro HL

                ;DIVHL
                ;Divide HL por DE utilizando la sustracción iterativa
                ;Prácticamente utiliza una ADD iterativa del complemento del divisor

                ;Parámetros de entrada
                ;       HL = dividendo
                ;       DE = divisor

                ;Parámetros de salida
                ;       BC = cociente
                ;       HL = resto
                DIVHL:
008F D5              PUSH    D          ;Guardar divisor
                                        ;Nota : El complemento a 2 se obtiene
                                        ;invirtiendo todos los bits y + 1
0090 7B              MOV     A,E        ;Complementar divisor (para hacer
0091 2F              CMA
0092 5F              MOV     E,A        ;después ADD iterativo)
0093 7A              MOV     A,D        ;Obtención byte MS
0094 2F              CMA                ;Complementación de éste
0095 57              MOV     D,A
0096 13              INX     D          ;Realización complemento a 2
                                        ;Restar el divisor negativo hasta que se haga
                                        ;negativo el dividendo contando el número
                                        ;de veces que se realiza la resta
0097 010000          LXI     B,0        ;Inicializar cociente
                DIVHLS:
009A 03              INX     B          ;Bucle de resta
009B 19              DAD     D          ;Suma de una unidad al cociente
009C DA9A00          JC      DIVHLS     ;Restar divisor
                                        ;Dividendo no negativo todavía
009F 0B              DCX     B          ;Dividendo ya negativo, cociente demasiado largo
                                        ;Cociente correcto
00A0 EB              XCHG               ;Cálculo resto correcto
00A1 E1              POP     H          ;DE = resto-divisor
00A2 19              DAD     D          ;Recuperar divisor positivo
00A3 C9              RET                ;HL = resto
                                        ;BC = cociente, HL = resto

                ;MULHL
                ;Producto HL * DE utilizando ADD iterativamente

                ;Parámetros de entrada
                ;       HL = multiplicando
                ;       DE = multiplicador

                ;Parámetro de salida
                ;       HL = producto
                ;       DE = multiplicador

                MULHL:
00A4 C5              PUSH    B          ;Guardar registro-de-usuario
                                        ;Comprobar si multiplicando
                                        ; o multiplicador nulos
```

Figura 5-24. (Continuación.)

```
00A5 7C              MOV     A,H
00A6 B5              ORA     L
00A7 CAC400          JZ      MULHLZ          ;Sí, producto ficticio
00AA 7A              MOV     A,D
00AB B3              ORA     E
00AC CAC400          JZ      MULHLZ          ;Sí, producto ficticio

                                             ;Esta rutina será más rápida si el
                                             ; menor valor se encuentra en DE
00AF 7A              MOV     A,D             ;Obtención de byte MS del valor DE en curso
00B0 BC              CMP     H               ;Comprobación de cuál es el menor
00B1 DAB500          JC      MULHLN          ;C a uno si D < H, por tanto no cambio
00B4 EB              XCHG

                MULHLN:
00B5 42              MOV     B,D             ;BC = multiplicador
00B6 4B              MOV     C,E
00B7 54              MOV     D,H             ;DE = HL = multiplicando
00B8 5D              MOV     E,L
00B9 0B              DCX     B               ;Ajustar conteo de manera que
                                             ;1 * multiplicando = multiplicando

                MULHLA:                      ;Bucle ADD
00BA 78              MOV     A,B             ;Comprobar si iteración completada
00BB B1              ORA     C
00BC CAC700          JZ      MULHLX          ;Sí, salida
00BF 19              DAD     D               ;HL = multiplicando + multiplicando
00C0 0B              DCX     B               ;Cuenta sobre multiplicador - 1
00C1 C3BA00          JMP     MULHLA          ;Vuelta atrás hasta hacer todas las ADD

                MULHLZ:
00C4 210000          LXI     H,0             ;Producto ficticio si multiplicando
                                             ; o multiplicador nulos

                MULHLX:
00C7 C1              POP     B               ;Recuperación registro-de-usuario
00C8 C9              RET

                ;SUBHL
                ;Sustracción HL - DE

                ;Parámetros de entrada
                ;       HL = minuendo
                ;       DE = sustraendo

                ;Parámetros de salida
                ;       HL = diferencia

                SUBHL:
00C9 7D              MOV     A,L             ;Obtención byte LS
00CA 93              SUB     E               ;Sustracción sin prestar atención al acarreo
00CB 6F              MOV     L,A             ;Situar en diferencia
00CC 7C              MOV     A,H             ;Obtención byte MS
00CD 9A              SBB     D               ;Sustracción incluyendo acarreo
00CE 67              MOV     H,A             ;Situar en diferencia
00CF C9              RET

                ;SHLR
                ;Desplazamiento de HL un lugar a la derecha (dividiendo HL por 2)

                ;Parámetros de entrada
                ;       HL = valor a desplazar

                ;Parámetros de salida
                ;       HL = valor/2

                SHLR:
00D0 B7              ORA     A               ;Borrado de acarreo
00D1 7C              MOV     A,H             ;Obtención byte MS
00D2 1F              RAR                     ;Séptimo bit puesto del acarreo
                                             ; bit 0 va al acarreo
00D3 67              MOV     H,A             ;Resituar MS byte desplazado
00D4 7D              MOV     A,L             ;Obtención byte LS
00D5 1F              RAR                     ;Bit 7 = bit 0 del byte MS
00D6 6F              MOV     L,A             ;Puesta en resultado
00D7 C9              RET
```

Figura 5-24. (Continuación.)

GAB (obtención del bloque de situación) es el inverso del GTAS (obtención de pista y sector). Devuelve el número de bloque, dados una pista y un sector.

Finalmente, la figura 5-24 incluye algunas subrutinas útiles de 16 bits para dividir el par de registro HL por DE (DIVHL), para multiplicar HL por DE (MULHL), para restar DE de HL (SUBHL —esto puede también utilizarse como comparación de 16 bits—) y para desplazar HL un bit a la derecha (SHLR). Las subrutinas de división y multiplicación son de alguna manera primitivas, utilizan iterativamente la sustracción y la adición respectivamente. No obstante, representan su papel como subrutinas de soporte.

Función 32: Fijación/obtención del número de usuario

Código de función: C=20H
Parámetros de entrada: E=0FFH para obtener el número de usuario, o
E=0 a 15 para fijar el número de usuario
Parámetros de salida: A=Número de usuario en curso si E era 0FFH

Ejemplo

```
0020 =          B$SETGETUN      EQU     32      ;Obtención/fijación número de usuario
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                                                ;Para fijar número de usuario
0000 0E20               MVI     C,B$SETGETUN    ;Código de función
0002 1E0F               MVI     E,15            ;Número de usuario requerido
0004 CD0500             CALL    BDOS            ;Para obtener el número de usuario
0007 0E20               MVI     C,B$SETGETUN    ;Código de función
0009 1EFF               MVI     E,0FFH          ;Indicar petición a GET
000B CD0500             CALL    BDOS            ;A = número usuario en curso (0 -- 15)
```

Propósito Esta subrutina lo mismo fija que obtiene los números de usuario en curso. Este número determina qué entradas al directorio se comparan durante todas las operaciones de fichero del disco.

Cuando se llama a esta función, los contenidos del registro E especifican cuál es la acción que se ha de realizar. Si E=0FFH, entonces la función devolverá el número de usuario en curso al registro A. Si se sitúa en E un número de 0 a 15 (es decir, un número de usuario válido), la función fijará el número de usuario actual a este valor.

Notas Puede utilizarse esta función para compartir ficheros con otros usuarios. Se puede localizar un fichero intentando abrir un fichero y conmutando a través de todos los números de usuario. O bien se puede compartir un fichero en otro número de usuario poniéndose en dicho número, operando en el fichero y volviendo entonces al número de usuario original.

Si se cambia el número de usuario en curso, habrá que proveer en el

programa el retorno al número original antes de que el programa termine. Es desconcertante para los operadores de computadoras encontrarse con que están en un número de usuario distinto después de un programa. De esta forma, los ficheros pueden ser dañados fácilmente o bien borrados.

Función 33: Lectura aleatoria

Código de función: C=21H
Parámetros de entrada: DE=Dirección del FCB
Parámetros de salida: A=Código de retorno

Ejemplo

```
0021 =          B$READRAN       EQU     33      ;Lectura aleatoria
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda en unidad de disco por defecto
0001 46494C454EFCB$NAME:        DB      'FILENAME'      ;Nombre de fichero
0009 545950     FCB$TYP:        DB      'TYP'   ;Tipo de fichero
000C 00         FCB$EXTENT:     DB      0       ;Extensión
000D 0000       FCB$RESV:       DB      0,0     ;Reservado para CP/M
000F 00         FCB$RECUSED:    DB      0       ;Registros utilizados en esta extensión
0010 0000000000FCB$ABUSED:      DB      0,0,0,0,0,0,0,0 ;Bloques utilizados
0018 0000000000                 DB      0,0,0,0,0,0,0,0
0020 00         FCB$SEQREC:     DB      0       ;Reg. de lec./escritura secuencial
0021 0000       FCB$RANREC:     DW      0       ;Reg. de lec./escritura aleatoria
0023 00         FCB$RANRECO:    DB      0       ;Byte de desbordamiento de registro
                                                ; aleatorio (MS)

0024 D204       RANRECNO:       DW      1234    ;Ejemplo de número de registro aleatorio

                                                ;El registro se leerá en la dirección
                                                ; fijada por la llamada previa
                                                ; a SETDMA
0026 2A2400             LHLD    RANRECNO        ;Obtención número de registro aleatorio
0029 222100             SHLD    FCB$RANREC      ;Activación bloque control de fichero
002C 0E21               MVI     C,B$READRAN     ;Código de función
002E 110000             LXI     D,FCB           ;DE -> bloque de control de fichero
0031 CD0500             CALL    BDOS            ;A = 00 si operación realizada con éxito
                                                ;A = 0 si el dato no está en el
                                                ; fichero. Específicamente:
                                                ;A = 01 -- intento de lectura de
                                                ;       registro no escrito
                                                ;    03 -- CP/M no puede cerrar
                                                ;       extensión en curso
                                                ;    04 -- intento de lectura de extensión
                                                ;       no escrita
                                                ;    06 -- intento de lectura después
                                                ;       del final del disco
```

Propósito Esta función lee un registro específico CP/M (128 bytes) de un fichero de acceso directo, es decir, un fichero en el que se tiene acceso directo a los registros. Supone que ya se ha abierto el fichero, fija la dirección de DMA utilizando la función correspondiente del BDOS y fija el registro específico a leer en el número de registro aleatorio del FCB. Esta función calcula la extensión del número de registro específico e intenta abrirlo y leer el registro correcto CP/M en la dirección de DMA.

El número de registro aleatorio del FCB tiene tres bytes de longitud (en

los bytes relativos 33, 34 y 35). El byte 33 es el menos significativo, el 34 el medio y el 35 el más significativo. El CP/M sólo utiliza el byte más significativo (35) para calcular el tamaño general del fichero (función 35). Es preciso poner a 0 este byte cuando se fija el FCB. Los bytes 33 y 34 se usan juntos para la lectura aleatoria, de forma que se puede acceder desde el registro 0 al 65.535 (un tamaño máximo de fichero de 8.388.480 bytes).

Esta función vuelve con A puesto a 0 para indicar que la operación se ha completado satisfactoriamente o con A puesto a un valor distinto a cero si ha ocurrido algún error. Los códigos de error son los siguientes:

A=01 (intento de leer un registro no escrito)

A=03 (el CP/M no ha podido cerrar la extensión en curso)

A=04 (intento de leer una extensión no escrita)

A=06 (intento de leer más allá del final del disco)

A diferencia de la función de lectura secuencial del BDOS (código 20, 14H), que actualiza el número de registro (secuencial) en curso en el FCB, la función de lectura aleatoria deja invariable el número de registro hasta que una nueva escritura aleatoria reemplace el registro que acaba de leerse.

Se puede seguir una lectura aleatoria con una escritura secuencial (código 21, 15H). Esta volverá a escribir el registro que se acaba de leer, pero después actualizará el número de registro secuencial. O se puede elegir utilizar la lectura secuencial después de la lectura aleatoria. En este caso, el mismo registro será releído y el número de registro secuencial aumentará. Abreviando, el fichero puede ser leído o escrito secuencialmente una vez que la lectura aleatoria se ha utilizado en la posición requerida en el fichero.

Notas

Para usar la función de lectura aleatoria, primero hay que abrir la *extensión de base* del fichero, es decir, la extensión 0. Aunque puede no haber registros de datos reales en esta extensión, su apertura permite que el fichero sea procesado correctamente.

Un problema que no se presenta de inmediato con los ficheros de acceso directo es que pueden ser creados fácilmente con espacios internos vacíos. Si se hubiera de crear el fichero con los registros números 0 y 5.000, no intervendría ninguna extensión de fichero. Si se intenta leer o copiar el fichero secuencialmente, incluso utilizando las utilidades de copiado de ficheros del CP/M, solamente se copiaría la primera extensión (en este caso, el registro 0). Una función de lectura secuencial devolvería un error de "final de fichero" después de haber leído el registro 0. Por consiguiente, hay que ser consciente del tipo de fichero que se intenta leer.

Véase la figura 5-26 como ejemplo de subrutina que realiza lecturas y escrituras en ficheros de acceso directo. Lee y escribe registros de tamaños

distintos a 128 bytes, donde es necesario leer o escribir varios registros del CP/M, leyéndolos previamente sobre su propia memoria intermedia cuando el registro que se escribe ocupa sólo parte del registro CP/M. Contiene también subrutinas para producir un producto de 32 bits que surge de multiplicar HL por DE (MLDL —multiplicación de doble longitud—) y un desplazamiento de un bit a la derecha para DE, HL (SDLR —desplazamiento a la derecha de doble longitud—).

Función 34: Escritura aleatoria

Código de función: C=22H
Parámetros de entrada: DE=Dirección del bloque de control de fichero
Parámetros de salida: A=Código de retorno

Ejemplo

```
0022 =          B$WRITERAN      EQU     34      ;Escritura aleatoria
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda en disco por defecto
0001 46494C454EFCB$NAME:        DB      'FILENAME'      ;Nombre de fichero
0009 545950     FCB$TYP:        DB      'TYP'   ;Tipo de fichero
000C 00         FCB$EXTENT:     DB      0       ;Extensión
000D 0000       FCB$RESV:       DB      0,0     ;Reservado para CP/M
000F 00         FCB$RECUSED:    DB      0       ;Registros utilizados en esta extensión
0010 0000000000FCB$ABUSED:      DB      0,0,0,0,0,0,0,0 ;Bloques utilizados
0018 0000000000                 DB      0,0,0,0,0,0,0,0
0020 00         FCB$SEQREC:     DB      0       ;Reg. lect./escrit. secuencial
0021 0000       FCB$RANREC:     DW      0       ;Reg. lect./escrit. aleatoria
0023 00         FCB$RANRECO:    DB      0       ;Byte de desbordamiento de registro
                                                ; aleatorio (MS)

0024 D204       RANRECNO:       DW      1234    ;Ejemplo de número de registro aleatorio

                                                ;El registro será escrito desde la
                                                ; dirección fijada por la llamada
                                                ; previa a SETDMA

0026 2A2400             LHLD    RANRECNO        ;Obtención número de fichero aleatorio
0029 222100             SHLD    FCB$RANREC      ;Activación bloque de control de
002C 0E22               MVI     C,B$WRITERAN    ; fichero
002E 110000             LXI     D,FCB           ;Código de función
0031 CD0500             CALL    BDOS            ;DE -> bloque de control de fichero
                                                ;A = 00 si operación terminada con éxito
                                                ;A = 0 si no hay datos en el fichero,
                                                ; concretamente:
                                                ;A = 03 -- CP/M no puede cerrar la
                                                ;       extensión en curso
                                                ;    05 -- directorio completo
                                                ;    06 -- intento de escritura
                                                ;       después del final del disco
```

Propósito Esta función escribe un registro específico del CP/M (128 bytes) en un fichero aleatorio. Se inicia casi igual que la función consorte, lectura aleatoria (código 33, 21H). Supone que se ha abierto ya el fichero, fija la dirección de DMA a la dirección de la memoria que contiene el registro a escribir en el disco y pone el número de registro aleatorio del FCB en el registro específico que se está escribiendo. Esta función calcula también la

extensión en que se encuentra el número de registro específico y abre la extensión (creándola si no existe ya). Los códigos de error devueltos a A por esta llamada son los mismos que los de la lectura aleatoria, con la adición del código de error 05, que indica directorio lleno.

Igual que la lectura aleatoria (pero contrariamente a la escritura secuencial), esta función no realiza una actualización de la extensión lógica y del número de registro secuencial (en curso) en el FCB. Por consiguiente, cualquier operación secuencial siguiente accederá al registro recién escrito por la llamada a lectura aleatoria, pero estas funciones actualizarán el número de registro secuencial. La escritura aleatoria puede utilizarse, por consiguiente, para situarse en el lugar requerido del fichero, al cual puede accederse entonces secuencialmente.

Notas

Con el fin de utilizar la escritura aleatoria, primero hay que abrir la extensión base (extensión 0) del fichero. Aunque pueden existir registros sin ningún dato en esta extensión, la apertura permite que el fichero sea procesado correctamente.

Como se ha explicado en las notas para la función de lectura aleatoria, puede crearse fácilmente un fichero aleatorio con espacios vacíos en su interior. Si se crea un fichero con número de registro 0 y número de registro 5.000, no habrá extensiones intermedias.

La figura 5-25 muestra un ejemplo de subrutina que crea un fichero aleatorio (CRF) pero evita este problema. Se debe especificar el número de registros de 128 bytes del CP/M del fichero. La subrutina crea éste y entonces escribe registros llenos de ceros en su totalidad. Esto hace más fácil procesar el fichero y permite a los programas de utilidad estándar del CP/M copiar el fichero porque hay un registro de datos en cada posición de grabación lógica del fichero. Ya no es un fichero "raro".

La figura 5-26 muestra una subrutina que liga las funciones de lectura y escritura aleatorias. Realiza las operaciones aleatorias (RO). A diferencia de las funciones estándar del BDOS que operan en registros CP/M de 128 bytes, el RO puede tratar tamaños de registro arbitrarios que van desde uno hasta varios miles de bytes. Se debe especificar el número de registro relativo del registro, no el del CP/M (el RO calcula este último). El RO lee también previamente un registro CP/M cuando el registro lógico a copiar ocupa parte de un registro de 128 bytes, bien porque es menor de 128 bytes o porque se extiende más de un sector de 128 bytes. La subrutina suprime esta lectura previa si se utiliza un tamaño de registro múltiplo de 128 bytes. En este caso, los registros encajarán exactamente en uno de 128 bytes, de forma que nunca existirá un sector de 128 bytes parcialmente ocupado.

Este ejemplo contiene también subrutinas para producir un producto de 32 bits, que surge de multiplicar HL por DE (MLDL —multiplicación de doble longitud—), y un desplazamiento de un bit a la derecha para DE, HL (SDLR —desplazamiento a la derecha de doble longitud—).

```
                        ;CRF
                        ;Creación de fichero de acceso directo (aleatorio)
                        ;Esta subrutina crea un fichero aleatorio. Antes de crearlo borra
                        ;cualquier otro que exista previamente y entonces rellena
                        ;con ceros el fichero completo.

                        ;Parámetros de entrada
                        ;       DE -> bloque de control para el nuevo fichero
                        ;       HL = número de registros CP/M de 128 bytes que deben
                        ;               rellenarse con ceros.

                        ;Parámetros de salida
                        ;       Acarreo a cero si la operación tiene éxito (A = 0,1,2,3)
                        ;       Acarreo a uno si error (A = 0FFH)

                        ;Secuencia de llamada
                        ;       LXI     D,FCB
                        ;       CALL    CRF
                        ;       JC      ERROR

0013 =          B$ERASE         EQU     19      ;Borrado de fichero
0016 =          B$CREATE        EQU     22      ;Creación de fichero
001A =          B$SETDMA        EQU     26      ;Fijación de dirección DMA
0015 =          B$WRITESEQ      EQU     21      ;Escritura de registro secuencial
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                CRFBUF:                         ;Memoria intermedia llena con ceros
0000 0000000000         DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                                  0,0,0
0032 0000000000         DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                                  0,0,0
0064 0000000000         DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

0080 0000       CRFRC:  DW      0               ;Contador de registros

                CRF:
0082 228000             SHLD    CRFRC           ;Guardar contador de registros
0085 D5                 PUSH    D               ;Conservar apuntador a FCB
0086 0E13               MVI     C,B$ERASE       ;Borrado de cualquier fichero
0088 CD0500             CALL    BDOS
008B D1                 POP     D               ;Recuperación de apuntador a FCB
008C D5                 PUSH    D               ; y volver a guardarlo
008D 0E16               MVI     C,B$CREATE      ;Creación y apertura de nuevo fichero
008F CD0500             CALL    BDOS
0092 FEFF               CPI     0FFH            ;Acarreo a 1 si OK, a 0 si error
0094 3F                 CMC                     ;Inversión para usar acarreo a 1 si error
0095 D1                 POP     D               ;Recuperar dirección del FCB
0096 D8                 RC                      ;Retorno si hay error
0097 D5                 PUSH    D               ;Volver a guardar apuntador a FCB

0098 0E1A               MVI     C,B$SETDMA      ;Fijación de dirección DMA a mem. int.-O
009A 110000             LXI     D,CRFBUF
009D CD0500             CALL    BDOS
00A0 D1                 POP     D               ;Recuperación apuntador a FCB

                CRFL:
00A1 2A8000             LHLD    CRFRC           ;Obtención cuenta de registros
00A4 7D                 MOV     A,L
00A5 B4                 ORA     H               ;Comprobación de cuenta a cero
00A6 C8                 RZ                      ;Sí, salida
00A7 2B                 DCX     H               ;Descontar unidad de contador
00A8 228000             SHLD    CRFRC           ;Almacenar contador
00AB D5                 PUSH    D               ;Volver a guardar apuntador a FCB
00AC 0E15               MVI     C,B$WRITESEQ    ;Escritura secuencial
00AE CD0500             CALL    BDOS

00B1 D1                 POP     D               ;Recuperación FCB
00B2 C3A100             JMP     CRFL            ;Escritura registro siguiente
```

Figura 5-25. Creación de fichero aleatorio.

```
                        ;RO
                        ;Operación de acceso directo (aleatoria) (lectura o escritura)

                        ;Esta subrutina escribe o lee un registro aleatorio de un fichero.
                        ;La longitud del registro puede ser distinta de 128 bytes. Esta
                        ;subrutina calcula el comienzo del registro CP/M (que es de 128 bytes)
                        ;y en el caso de lectura, realiza una lectura aleatoria y traslada
                        ;el registro especificado a una memoria intermedia del usuario. Si es necesario,
                        ;se leerán más registros CP/M hasta que el registro especificado por el usuario
                        ;haya sido leído completamente. En escritura, si el tamaño de registro
                        ;especificado por el usuario no es múltiplo exacto de registros C,
                        ;serán preleídos los sectores apropiados. Esta prelectura no es necesaria
                        ;cuando el registro especificado por el usuario es un número exacto
                        ;de registros CP/M ni cuando la subrutina procesa registros CP/M que cubren
                        ;completamente el registro especificado por el usuario.

                        ;Parámetros de entrada
                        ;       HL -> bloque de parámetros de la forma:
                        ;               DB      0       ;OFFH para lectura, 00H para escritura
                        ;               DW      FCB     ;Apuntador al FCB
                        ;               DW      RECNO   ;Número de registros usuario
                        ;               DW      RECSZ   ;Tamaño de registro usuario
                        ;               DW      BUFFER  ;Apuntador a la memoria intermedia del usuario
                        ;                               ; de longitud RECSZ bytes

                        ;Parámetros de salida
                        ;       A = 0 si completa la operación (registro usuario
                        ;               copiado en memoria intermedia del usuario)
                        ;           1 Intento de lectura de registro CP/M no escrito
                        ;           3 CP/M no puede cerrar una extensión
                        ;           4 Intento de lectura de extensión no escrita
                        ;           5 CP/M no puede crear una nueva extensión
                        ;           6 Intento de lectura después de final de disco

                        ; Secuencia de llamada
                        ;       LXI     H,PARAMS        ;HL -> bloque de parámetros
                        ;       CALL    RO
                        ;       ORA     A               ;Comprobación de error
                        ;       JNZ     ERROR

0021 =          FCBE$RANREC     EQU     33      ;Desplazamiento de número de registro aleatorio en FCB
001A =          B$SETDMA        EQU     26      ;Fijación de dirección DMA
0021 =          B$READRAN       EQU     33      ;Lectura de registro aleatorio
0028 =          B$WRITERANZ     EQU     40      ;Escritura de registro aleatorio con rellenado
                                                ; de ceros en bloques desubicados
                                                ; previamente
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                ROPB:                           ;Imagen del bloque de par.
0000 00         ROREAD: DB      0               ;NZ en lectura, Z en escritura
0001 0000       ROFCB:  DW      0               ;Apuntador al FCB
0003 0000       ROURN:  DW      0               ;Número de registro usuario
0005 0000       ROURL:  DW      0               ;Longitud de registro usuario
0007 0000       ROUB:   DW      0               ;Apuntador a memoria intermedia de usuario
0009 =          ROPBL   EQU     $-ROPB          ;Longitud bloque de parámetros

0009 0000       ROFRP:  DW      0               ;Apuntador al comienzo del fragmento de registro usuario
                                                ; en primer registro CP/M leído
000B 00         ROFRL:  DB      0               ;Longitud del fragmento
000C 0000       RORNP:  DW      0               ;Apuntador a número de registro (en FCB de usuario)
000E 00         ROWECR: DB      0               ;NZ cuando se trata de esc. de registro que son un
                                                ; número exacto de registros CP/M (y que
                                                ; por tanto no necesitan prelectura)

000F            ROBUF:  DS      128             ;Memoria intermedia para registro CP/M

                RO:
008F 110000             LXI     D,ROPB          ;DE -> bloque de parámetros locales
0092 0E09               MVI     C,ROPBL         ;Longitud del bloque de parámetros
0094 CDFE01             CALL    MOVE            ;Transferencia de C bytes de HL a DE

                        ;Para calcular el desplazamiento de registros usuario en registro
                        ; CP/M, calcular el byte de desplazamiento relativo del comienzo
                        ; del registro usuario en el fichero (por ejemplo, número de registro
                        ; usuario * tamaño del registro). Los 7 bits menos significativos
```

Figura 5-26. Lectura/escritura aleatoria de registros de longitud variable.

```
                        ; de este producto dan el byte de desplazamiento
                        ; del comienzo del registro usuario.
                        ;El producto / 128 (desplazado 7 bits) da el número de registro
                        ;CP/M del comienzo del registro usuario.

0097 2A0500             LHLD    ROURL           ;Obtención de longitud registro de usuario
009A 7D                 MOV     A,L             ;Obtención de los bytes LS de la longitud del reg. usuario
009B E67F               ANI     7FH             ;Comprobación de múltiplo exacto de 128
009D B7                 ORA     A               ;(es decir, número exacto de registros C)
009E 3E00               MVI     A,0             ;A = 0 no cambian los indicadores
00A0 C2A400             JNZ     RONE            ;No número exacto de registros CP
00A3 3D                 DCR     A               ;A = FF

                RONE:.
00A4 320E00             STA     ROWECR          ;Puesta a uno del indicador de esc.-de-núm.-exac.-reg.-C
00A7 EB                 XCHG                    ;DE = longitud de registro de usuario
00A8 2A0300             LHLD    ROURN           ;Obtención número de registro de usuario
00AB CDB801             CALL    MLDL            ;DE,HL = HL * DE
                                                ;DE,HL = bytes de desplaz. de reg. usuario en el fichero
00AE D5                 PUSH    D               ;Almacenar byte de desplazamiento de registro usuario
00AF E5                 PUSH    H
00B0 7D                 MOV     A,L             ;Obtención del byte LS del producto
00B1 E67F               ANI     7FH             ;Aislamiento de byte de desplazamiento en el

00B3 4F                 MOV     C,A             ;registro CP/M
00B4 0600               MVI     B,0             ;Situar como valor en palabra
00B6 210F00             LXI     H,ROBUF         ;Obtención de dirección base memoria intermedia local
00B9 09                 DAD     B               ;HL -> comienzo del fragmento en la memoria intermedia
00BA 220900             SHLD    ROFRP           ;Almacenar apuntador a fragmento

                        ;Calcular longitud máxima del fragmento que puede residir
                        ;en un registro CP/M basado en el byte de desplazamiento
                        ;del registro CP/M donde comienza el fragmento.

00BD 47                 MOV     B,A             ;Tomar copia del desplazamiento en el registro CP/M
00BE 3E80               MVI     A,128           ;Tamaño de registro CP/M
00C0 90                 SUB     B               ;Cálculo de byte de desplazamiento 128
00C1 320B00             STA     ROFRL           ;Suposición de que es la longitud del fragmento

                        ;Si la longitud de registro del usuario es menor que la supuesta,
                        ; se usará ésta en lugar del resultado anterior

00C4 47                 MOV     B,A             ;Obtención de copia de long. de fragmento supuesta
00C5 3A0600             LDA     ROURL+1         ;Obtención del MS byte de la long. del reg. de usuario
00C8 B7                 ORA     A               ;Si NZ, long. de reg. debe ser > 128
00C9 C2D600             JNZ     ROFLOK          ;En este caso long. de fragmento OK
00CC 3A0500             LDA     ROURL           ;Todavía es posible que la long. del frag.
00CF B8                 CMP     B               ; sea menor que la long. del reg.
00D0 D2D600             JNC     ROFLOK          ;NC si long. reg. usuario => long. fragmento
00D3 320B00             STA     ROFRL           ;Long. reg. de usuario < long. frag.
                                                ; puesta de la longitud de frag. al menor

                ROFLOK:
00D6 3A0E00             LDA     ROWECR          ;Obtención de indicador de reg. exacto CP/M
00D9 47                 MOV     B,A             ;para efectuar AND con indicador READ
00DA 3A0000             LDA     OREAD           ;Obtención indicador de operación de lectura (READ)
00DD 2F                 CMA                     ;Inversión de NZ si escritura
00DE A0                 ANA     B               ;Formación del AND lógico
00DF 320E00             STA     ROWECR          ;Guardar en indicador

                        ;Recuperación del byte de desplazamiento de doble longitud en el fichero
                        ;de comienzo del registro usuario. Desplazar 7 posiciones
                        ;a la derecha para dividir por 128 y obtener el número de registro CP/M
                        ;para el comienzo del registro usuario.

00E2 E1                 POP     H               ;Recuperar byte de desplazamiento del registro usuario
00E3 D1                 POP     D
00E4 0E07               MVI     C,7             ;Cuenta para desplazamiento a derecha

                ROS:
00E6 CDF101             CALL    SDLR            ;DE,HL = DE,HL / 2
00E9 0D                 DCR     C
00EA C2E600             JNZ     ROS
```

Figura 5-26. (Continuación.)

```
00ED 7A             MOV     A,D             ;Error si DE todavía NZ después
00EE B3             ORA     E               ; de dividir por 128
00EF C2AC01         JNZ     ROERO

                                            ;Activación del número de registro CP/M en el FCB
00F2 EB             XCHG                    ;DE = número de registro CP/M
00F3 2A0100         LHLD    ROFCB           ;Obtención de apuntador a FCB
00F6 012100         LXI     B,FCBE$RANREC   ;Desplazamiento de núm. de registro aleatorio en el FCB
00F9 09             DAD     B               ;HL -> número de registro aleatorio en el FCB
00FA 220C00         SHLD    RORNP           ;Almacenamiento de apuntador a número de registro
00FD 73             MOV     M,E             ;Almacenamiento de byte LS
00FE 23             INX     H
00FF 72             MOV     M,D             ;Almacenamiento de byte MS

0100 0E1A           MVI     C,B$SETDMA      ;Fijación dirección DMA a la memoria intermedia local
0102 110F00         LXI     D,ROBUF
0105 CD0500         CALL    BDOS

0108 3A0E00         LDA     'ROWECR         ;Ignorar prelectura si se escriben sectores exactos
010B B7             ORA     A
010C C21F01         JNZ     ROMNF

010F 2A0100         LHLD    ROFCB           ;Obtención apuntador a FCB
0112 EB             XCHG                    ;DE -> FCB
0113 0E21           MVI     C,B$READRAN     ;Función de lectura aleatoria
0115 CD0500         CALL    BDOS

0118 FE05           CPI     5               ;Comprobación de código de error < 5
011A DCAF01         CC      ROCIE           ;Sí, comprobar si error ignorable
                                            ; (es decir, error de lectura de fichero
                                            ; en la op. de prelectura)
011D B7             ORA     A               ;Comprobación de error
011E C0             RNZ                     ;Si

                ROMNF:                      ;Transferir nuevo fragmento
011F 2A0700         LHLD    ROUB            ;Obtención apuntador a memoria intermedia de usuario
0122 EB             XCHG                    ;DE -> memoria intermedia de usuario
0123 2A0900         LHLD    ROFRP           ;HL -> comienzo de registro usuario en mem. int. local
0126 3A0B00         LDA     ROFRL           ;Obtención de longitud de fragmento
0129 4F             MOV     C,A             ;Preparado para transferencia

012A 3A0000         LDA     ROREAD          ;Comprobación de lectura
012D B7             ORA     A
012E C23201         JNZ     RORD1           ;Sí, DE y HL permanecen sin cambio
0131 EB             XCHG                    ;Escritura, intercambiar fuente y destino
                                            ;DE -> comienzo registro de usuario en mem. int. local
                                            ;HL -> memoria intermedia de usuario
                RORD1:
0132 CDFE01         CALL    MOVE            ;Lectura - fragmento local -> mem. int. usuario
                                            ;Escritura - fragmento usuario -> mem. int. local
0135 3A0000         LDA     ROREAD          ;Comprobación de escritura
0138 B7             ORA     A
0139 CA3D01         JZ      ROWR1           ;Escritura, se mantiene HL -> mem. int. usuario
013C EB             XCHG                    ;HL -> byte siguiente en mem. int. de usuario
                ROWR1:
013D 220700         SHLD    ROUB            ;Guardar apuntador a mem. int. de usuario actualizada
0140 3A0000         LDA     ROREAD          ;Comprobación de lectura
0143 B7             ORA     A
0144 C25001         JNZ     RORD3           ;Sí, ignorar código de escritura

0147 0E28           MVI     C,B$WRITERANZ   ;Escritura aleatoria
0149 2A0100         LHLD    ROFCB           ;Obtención dirección del FCB
014C EB             XCHG                    ;DE -> FCB
014D CD0500         CALL    BDOS

                RORD3:  ;Calcular la longitud residual del registro usuario no transferido.
                        ;Si es necesario (a causa de que hay que transferir
                        ;una gran cantidad de datos) serán leídos más registros CP/M.
                        ;En este caso, el comienzo del fragmento de ser desplazado a 0.
                        ;La longitud del fragmento depende de cómo termine el registro
                        ;usuario en el siguiente registro o deba expandirse. Si la longitud
                        ;residual del registro de usuario es > 128, la longitud del fragmento se
                        ;pondrá a 128.
```

Figura 5-26. (Continuación.)

```
0150 2A0500             LHLD    ROURL       ;Obtención longitud residual de registro de usuario
0153 3A0B00             LDA     ROFRL       ;Obtención de la long. del frag. que acaba de transferirse
0156 5F                 MOV     E,A         ;Situación como valor en palabra
0157 1600               MVI     D,0
0159 CDEA01             CALL    SUBHL       ;Calcular ROURL - ROFRL
015C 7C                 MOV     A,H         ;Comprobación de resultado 0
015D B5                 ORA     L
015E C8                 RZ                  ;Retorno cuando se complete la transferencia
                                            ; del registro usuario
015F 220500             SHLD    ROURL       ;Guardar la longitud residual del registro ya descontada
0162 4D                 MOV     C,L         ;Suposición de longitud residual < 128
0163 118000             LXI     D,128       ;Comprobación longitud residual < 128
0166 CDEA01             CALL    SUBHL       ;HL = HL - DE
0169 FA6E01             JM      ROLT128     ;Negativo si < 128
016C 0E80               MVI     C,128       ;=> 128, en consecuencia puesta a 128 de la long. de frag.

                ROLT128:
016E 79                 MOV     A,C
016F 320B00             STA     ROFRL       ;La longitud del fragmento es ahora 128
                                            ; si faltan más de 128 bytes para entrar
                                            ; en el registro usuario o exactamente
                                            ; el número de bytes correcto (< 128)
                                            ; para completar el registro de usuario
0172 210F00             LXI     H,ROBUF     ;Todos los registros CP/M siguientes comenzarán al principio
0175 220900             SHLD    ROFRP       ; de la memoria intermedia

                                            ;Actualización del número de registro aleatorio en el FCB
0178 2A0C00             LHLD    RORNP       ;HL -> número de registro aleatorio en FCB del usuario
017B 5E                 MOV     E,M         ;Incremento del número de registro aleatorio
017C 23                 INX     H           ;HL -> byte MS del número de registro
017D 56                 MOV     D,M         ;Obtención del byte MS
017E 13                 INX     D           ;Actualizar el número de registro
017F 7A                 MOV     A,D         ;Comprobación de número registro 0
0180 B3                 ORA     E
0181 C28701             JNZ     ROSRN       ;No, guardar número de registro
0184 3E06               MVI     A,6         ;Indicar "final del disco"
0186 C9                 RET                 ;Retorno al usuario
                ROSRN:
0187 72                 MOV     M,D         ;Almacenar número de registro
0188 2B                 DCX     H           ;HL -> byte LS
0189 73                 MOV     M,E

                                            ;En lectura, comprobar necesidad de prelectura
018A 3A0E00             LDA     ROWECR      ;Comprobación de escritura de núm. exacto de registro CP/M
018D B7                 ORA     A
018E C21F01             JNZ     ROMNF       ;Sí, transferencia fragmento siguiente

0191 3A0000             LDA     ROREAD      ;Si lectura, realizar una lectura incondicional
0194 B7                 ORA     A
0195 C2A001             JNZ     RORD2

0198 3A0B00             LDA     ROFRL       ;Para escritura, ignorar prelectura si todo
019B FE80               CPI     128         ; el registro CP/M ha sido reescrito
019D CA1F01             JZ      ROMNF       ; (longitud del fragmento = 128)

                RORD2:
01A0 0E21               MVI     C,B$READRAN ;Lectura del siguiente registro CP/M
01A2 2A0100             LHLD    ROFCB       ; en secuencia

01A5 EB                 XCHG                ;DE -> FCB
01A6 CD0500             CALL    BDOS
01A9 C31F01             JMP     ROMNF       ;Vuelta atrás para transferir fragmento siguiente

                ROERO:                      ;Error debido a número de registro usuario
                                            ; * longitud registro de usuario / 128 da
                                            ; un número de registros CP/M > 65.535
01AC 3E04               MVI     A,4         ;Indicación "intento de lectura de extensión
01AE C9                 RET                 ; no escrita"

                ROCIE:                      ;Comprobación error ignorable (prelectura
                                            ; para operación de escritura)
01AF 47                 MOV     B,A         ;Almacenamiento código de error original
01B0 3A0000             LDA     ROREAD      ;Comprobación de operación de lectura
01B3 B7                 ORA     A
01B4 78                 MOV     A,B         ;Restauración código de error original,
                                            ; pero permaneciendo los indicadores
```

Figura 5-26. (Continuación.)

```
01B5 C0                 RNZ                     ;Retorno si lectura
01B6 AF                 XRA     A               ;Indicador de no error ficticio
01B7 C9                 RET

                ;MLDL
                ;Multiplicación HL * DE utilizando ADD iterativo
                ;con devolución del producto en DE,HL.

                ;Parámetros de entrada
                ;       HL = multiplicando
                ;       DE = multiplicador

                ;Parámetros de salida
                ;       DE,HL = producto
                ;       DE = multiplicador

                MLDL:
01B8 010000             LXI     B,0             ;0 en cabecera de pila
01BB C5                 PUSH    B               ; para actuar como byte MS del producto
                                                ;Comprobación de multiplicando o
                                                ; multiplicador nulo
01BC 7C                 MOV     A,H
01BD B5                 ORA     L
01BE CAE501             JZ      MLDLZ           ;Sí, producto ficticio
01C1 7A                 MOV     A,D
01C2 B3                 ORA     E
01C3 CAE501             JZ      MLDLZ           ;Sí, producto ficticio

                                                ;Esta rutina será más rápida si el menor valor
                                                ; está en DE
01C6 7A                 MOV     A,D             ;Obtención del byte MS del valor DE en curso
01C7 BC                 CMP     H               ;Comprobación del menor
01C8 DACC01             JC      MLDLNX          ;C activado si D < H, en consecuencia no cambia
01CB EB                 XCHG
                MLDLNX:
01CC 42                 MOV     B,D             ;BC = multiplicador
01CD 4B                 MOV     C,E

01CE 54                 MOV     D,H             ;DE = HL = multiplicando
01CF 5D                 MOV     E,L

01D0 0B                 DCX     B               ;Ajustar contador de forma que
                                                ; 1 * multiplicando = multiplicando
                MLDLA:                          ;Bucle ADD
01D1 78                 MOV     A,B             ;Comprobación si completadas todas las iteraciones
01D2 B1                 ORA     C
01D3 CAE801             JZ      MLDLX           ;Sí, salida
01D6 19                 DAD     D               ;HL = multiplicando + multiplicando
01D7 E3                 XTHL                    ;HL = byte MS del resultado, TOS = prod. parcial
01D8 7D                 MOV     A,L             ;Obtención byte LS de primera mitad del producto
01D9 CE00               ACI     0               ;Sumar 1 si acarreo activo
01DB 6F                 MOV     L,A             ;Reemplazamiento
01DC 7C                 MOV     A,H             ;Repetir para byte MS
01DD CE00               ACI     0                       /
01DF 67                 MOV     H,A
01E0 E3                 XTHL
01E1 0B                 DCX     B               ;Cuenta atrás del multiplicador-1
01E2 C3D101             JMP     MLDLA           ;Bucle atrás hasta hacer todos los ADDs
                MLDLZ:
01E5 210000             LXI     H,0             ;Simulación del producto si multiplicando o
                                                ; multiplicador son nulos

                MLDLX:
01E8 D1                 POP     D               ;Recuperación parte MS del producto
01E9 C9                 RET


                ;SUBHL
                ;Sustracción HL - DE

                ;Parámetros de entrada
                ;       HL = minuendo
                ;       DE = sustraendo

                ;Parámetros de salida
                ;       HL = diferencia
```

Figura 5-26. (Continuación.)

```
                SUBHL:
01EA 7D                 MOV     A,L             ;Obtención byte LS
01EB 93                 SUB     E               ;Sustracción sin tener en cuenta el acarreo
01EC 6F                 MOV     L,A             ;Puesta en la diferencia
01ED 7C                 MOV     A,H             ;Obtención byte MS
01EE 9A                 SBB     D               ;Sustracción incluido el acarreo
01EF 67                 MOV     H,A             ;Transferir a diferencia
01F0 C9                 RET

                ;SDLR
                ;Desplazamiento de DE y HL un lugar a la derecha (dividiendo DE y HL por dos)

                ;Parámetros de entrada
                ;       DE,HL = valor a desplazar

                ;Parámetros de salida
                ;       DE,HL = valor / 2

                SDLR:
01F1 B7                 ORA     A               ;Borrar acarreo
01F2 EB                 XCHG                    ;Desplazar primero DE
01F3 CDF701             CALL    SDLR2
01F6 EB                 XCHG                    ;Desplazar HL

                                                ;Salto a SDLR2 con acarreo puesto correctamente
                                                ; a partir del bit LS
                                                ; de DE
                SDLR2:                          ;Desplazamiento de HL un lugar a la derecha
01F7 7C                 MOV     A,H             ;Obtención byte MS
01F8 1F                 RAR                     ;Bit 7 a 1 del acarreo previo
                                                ;Bit 0 entra en acarreo
01F9 67                 MOV     H,A             ;Puesta del byte MS desplazado
01FA 7D                 MOV     A,L             ;Obtención byte LS
01FB 1F                 RAR                     ;Bit 7 = bit 0 de byte MS
01FC 6F                 MOV     L,A             ;Puesta en resultado
01FD C9                 RET

                ;MOVE
                ;Transferir C bytes desde HL a DE

                MOVE:
01FE 7E                 MOV     A,M             ;Obtención byte fuente
01FF 12                 STAX    D               ;Almacenar en destino
0200 13                 INX     D               ;Actualización apuntador destino
0201 23                 INX     H               ;Actualización apuntador fuente
0202 0D                 DCR     C               ;Decremento del contador
0203 C2FE01             JNZ     MOVE            ;Obtención byte siguiente
0206 C9                 RET
```

Figura 5-26. (Continuación.)

Función 35: Obtención del tamaño de fichero

Código de función: C=23H
Parámetros de entrada: DE=Dirección del FCB
Parámetros de salida: Campo de registro aleatorio fijado en el FCB

Ejemplo

```
0023 =          B$GETFSIZ       EQU     35      ;Obtención tamaño fichero lóg. aleatorio
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda de unidad de disco por defecto
0001 46494C454EFCB$NAME:        DB      'FILENAME'      ;Nombre de fichero
0009 545950     FCB$TYP:        DB      'TYP'   ;Tipo de fichero
000C 00         FCB$EXTENT:     DB      0       ;Extensión
000D 0000       FCB$RESV:       DB      0,0     ;Reservado para CP/M
```

```
000F 00          FCB$RECUSED:    DB      0       ;Registros usados en esta extensión
0010 0000000000FCB$ABUSED:       DB      0,0,0,0,0,0,0,0  ;Bloques utilizados
0018 0000000000                  DB      0,0,0,0,0,0,0,0
0020 00          FCB$SEQREC:     DB      0       ;Reg. de lectura/escritura secuencial
0021 0000        FCB$RANREC:     DW      0       ;Reg. de lectura/escritura aleatoria
0023 00          FCB$RANRECO:    DB      0       ;Byte de desbordamiento de registro
                                                 ; aleatorio (MS)

0024 0E23              MVI       C,B$GETFSIZ     ;Código de función
0026 110000            LXI       D,FCB           ;DE -> bloque de control de fichero
0029 CD0500            CALL      BDOS
002C 2A2100            LHLD      FCB$RANREC      ;Obtención núm. de registro aleatorio
                                                 ;HL = tamaño del fichero lógico
                                                 ; es decir, número de registro del
                                                 ; último registro
```

Propósito Esta función devuelve el tamaño virtual del fichero especificado. Lo hace fijando el número de registro aleatorio (bytes 33-35), en el FCB específico, al máximo número de registros de 128 bytes del fichero. El tamaño virtual del fichero se calcula desde la dirección del registro hasta el final de fichero. Los bytes 33 y 34 forman un valor de 16 bits, que contiene el número de registro cuyo desbordamiento se indica en el byte 35. Si el byte 35 es 01, esto significa que el fichero tiene una cantidad máxima de registros de 65.536.

Si la función no puede encontrar el fichero especificado en el FCB, vuelve con el campo de registro aleatorio puesto a cero.

Puede usarse esta función cuando se quieran añadir datos al final de un fichero existente. Llamando primero a esta función, los bytes de registro aleatorio se pondrán al final del fichero. Las llamadas siguientes a escritura aleatoria escribirán registros en esta dirección preestablecida.

Notas No se debe confundir el tamaño virtual del fichero con el tamaño del fichero real. En un fichero aleatorio, si se escribe únicamente un registro CP/M en el número de registro 1.000 y entonces se llama a esta función, volverá con el campo de número de registro en el FCB puesto a 1.000 —aun cuando exista un solo registro en el fichero.

Para ficheros secuenciales, esta función devuelve el número de registros del fichero. En este caso, los tamaños de los ficheros virtual y actual coinciden.

Función 36: Fijación del número de registro aleatorio

Código de función: C=24H
Parámetros de entrada: DE=Dirección del FCB
Parámetros de salida: Campo de registro aleatorio fijado en el FCB

Ejemplo

```
0024 =          B$SETRANREC     EQU     36      ;Fijación número de registro aleatorio
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS
```

```
                        FCB:
0000 00                 FCB$DISK:       DB      0           ;Bloque de control de fichero
0001 46494C454E         FCB$NAME:       DB      'FILENAME'  ;Búsqueda de unidad de disco por defecto
0009 545950             FCB$TYP:        DB      'TYP'       ;Nombre de fichero
000C 00                 FCB$EXTENT:     DB      0           ;Tipo de fichero
000D 0000               FCB$RESV:       DB      0,0         ;Extensión
000F 00                 FCB$RECUSED:    DB      0           ;Reservado para CP/M
0010 00000000000        FCB$ABUSED:     DB      0,0,0,0,0,0,0,0,0,0 ;Registros usados en esta extensión
0018 00000000000                        DB      0,0,0,0,0,0,0,0,0,0 ;Bloques utilizados
0020 00                 FCB$SEQREC:     DB      0           ;Reg. de lectura/escritura secuencial
0021 0000               FCB$RANREC:     DW      0           ;Reg. de lectura/escritura aleatoria
0023 00                 FCB$RANRECO:    DB      0           ;Byte desbordamiento de registro
                                                            ; aleatorio (MS)

                                                            ;... fichero abierto y leído
                                                            ; o escrito secuencialmente...

0024 0E24               MVI     C,B$SETRANREC   ;Código de función
0026 110000             LXI     D,FCB           ;DE -> bloque de control de fichero
0029 CD0500             CALL    BDOS
002C 2A2100             LHLD    FCB$RANREC      ;Obtención núm. de registro aleatorio
                                                ;HL = número de registro aleatorio
                                                ; corresponde al avance
                                                ; hacia abajo del
                                                ; fichero
```

Propósito Esta función fija el número de registro aleatorio del FCB en el valor correcto para el último registro leído o escrito secuencialmente en el fichero.

Notas Esta función suministra una forma conveniente para construir ficheros de índices, de forma que se pueda acceder aleatoriamente a un fichero secuencial. Abre el fichero secuencial y, cuando se lee cada registro, extrae el campo de claves apropiado de los registros de datos. Hace la petición de fijación de registro aleatorio al BDOS y crea un nuevo registro de datos justo con el campo de claves y el número de registro aleatorio. Escribe el nuevo registro de datos en el fichero de índices.

Una vez que se ha hecho esto para cada registro del fichero, el fichero de índices significa un método conveniente, dado un valor de la clave de búsqueda, para encontrar el registro CP/M en el que se encuentran los datos.

También se puede usar esta función como medio de averiguar la posición en curso en un fichero secuencial —bien para relacionar un número de registro CP/M con la posición, o simplemente para marcar el sitio de forma que permita volver a situarse más tarde en el mismo lugar.

Función 37: Puesta a cero del controlador de disco lógico

Código de función: C=25H
Parámetros de entrada: DE=Mapa de bits del controlador lógico
Parámetros de salida: A=00H

Ejemplo

```
0025 =          B$RESETD        EQU     37      ;Puesta a 0 discos lógicos
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS
                                                ;DE = mapa de bits de los discos
                                                ; a poner a 0
                                                ;Los bits son iguales a 1 si el disco
                                                ; debe ser puesto a 0
                                                ;Bits 15 14 13 ... 2 1 0
                                                ;Disco P  0  N ... C B A
0000 110200             LXI     D,0000$0000$0000$0010B ;Puesta a 0 unidad B:
0003 0E25               MVI     C,B$RESETD      ;Código de función
0005 CD0500             CALL    BDOS
```

Propósito Esta función pone a cero los controladores individuales de disco. Es una versión más precisa de la función de puesta a cero del sistema de disco (código 13, 0DH), en la que pueden fijarse discos lógicos específicos mejor que todos ellos.

El mapa de bits de DE muestra qué discos han de ponerse a cero. El bit menos significativo de E representa el disco A, y el bit más significativo de D el disco P. Los bits puestos a 1 indican que los discos se han de poner a cero.

Obsérvese que esta función devuelve un valor cero en A para mantener la compatibilidad con el MP/M.

Notas Esta función deberá utilizarse sólo cuando los disquetes especificados necesiten cambiarse. El cambio de un disquete sin una petición al CP/M para que lo registre provocará que el BDOS suponga que se ha cometido un error y coloque el nuevo disco en modo de sólo lectura como medida de protección.

Función 40: Escritura aleatoria con rellenado de ceros

Código de función: C = 28H
Parámetros de entrada: DE = Dirección del FCB
Parámetros de salida: A = Código de retorno

Ejemplo

```
0028 =          B$WRITERANZ     EQU     40      ;Escritura aleatoria con rellenado
                                                ; de ceros
0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS

                FCB:                            ;Bloque de control de fichero
0000 00         FCB$DISK:       DB      0       ;Búsqueda de unidad de disco por defecto
0001 46494C454E FCB$NAME:       DB      'FILENAME'      ;Nombre de fichero
0009 545950     FCB$TYP:        DB      'TYP'   ;Tipo de fichero
000C 00         FCB$EXTENT:     DB      0       ;Extensión
000D 0000       FCB$RESV:       DB      0,0     ;Reservado para CP/M
000F 00         FCB$RECUSED:    DB      0       ;Registros usados en esta extensión
0010 00000000000FCB$ABUSED:     DB      0,0,0,0,0,0,0,0 ;Bloques utilizados
0018 0000000000                 DB      0,0,0,0,0,0,0,0
```

```
0020 00        FCB$SEQREC:     DB      0       ;Reg. de lectura/escritura secuencial
0021 0000      FCB$RANREC:     DW      0       ;Reg. de lectura/escritura aleatoria
0023 00        FCB$RANRECO:    DB      0       ;Byte desbordamiento de registro
                                               ; aleatorio (MS)

0024 D204      RANRECNO:       DW      1234    ;Ejemplo número de registro aleatorio

                                               ;El registro será escrito desde la
                                               ; dirección fijada por la llamada
                                               ; a SETDMA anterior
0026 2A2400            LHLD    RANRECNO        ;Obtención número registro aleatorio
0029 222100            SHLD    FCB$RANREC      ;Activación bloque de control de fichero
002C 0E28              MVI     C,B$WRITERANZ   ;Código de función
002E 110000            LXI     D,FCB           ;DE -> bloque de control de fichero
0031 CD0500            CALL    BDOS            ;A = 00 si operación terminada con éxito
                                               ;A = distinto de 0 si no hay datos en
                                               ; fichero. Concretamente:
                                               ;A = 03 -- CP/M no puede cerrar
                                               ;         extensión en curso
                                               ;    05 -- directorio completo
                                               ;    06 -- intento de escritura
                                               ;         después de final de disco
```

Propósito Esta función es una extensión de la función de escritura aleatoria descrita previamente. Además de realizar la escritura aleatoria, llenará cada nuevo bloque con 00H. Digital Research añadió esta función para asistir a Microsoft en la producción de su compilador COBOL —hace más fácil la lógica de manejo del código de fichero—. Además es una forma económica de llenar completamente un fichero aleatorio con 00H. Sólo se necesita escribir un registro por bloque; el BDOS borrará el resto del bloque.

Notas Consúltese la descripción de la función de escritura aleatoria (código 34).

Componentes del BIOS
Puntos de entrada del BIOS
Funciones de arranque y carga
Funciones de entrada/salida de caracteres
Funciones del disco
Llamada a las funciones del BIOS directamente
Ejemplo de BIOS

El sistema básico de entrada/salida

Este capítulo examina más detenidamente el sistema básico de entrada/salida (BIOS). El BIOS procura el software de conexión entre el procesador de órdenes de consola (CCP), el sistema operativo básico de disco (BDOS) y el hardware físico de la computadora. El CCP y el BDOS interaccionan con las partes del sistema de cálculo sólo como dispositivos lógicos. Por tanto, pueden permanecer invariables de un sistema a otro. El BIOS, sin embargo, se adapta al tipo particular de computadora y de controlador de disco. La única parte predecible del BIOS es la forma en que se encara con el CCP y el BDOS. Esto debe permanecer inalterable, sin tener en cuenta los dispositivos especiales que se hayan construido en el BIOS.

Componentes del BIOS

Un BIOS estándar consiste en subrutinas de bajo nivel que controlan cuatro tipos de dispositivos físicos:

- Consola: el CP/M se comunica con el mundo exterior por medio de la consola. Normalmente será un terminal de vídeo o un terminal de copia dura.
- "Lectora" y "perforadora": Estos dispositivos se utilizan normalmente para comunicarse entre computadoras —los nombres "lectora" y "perforadora" son reliquias históricas de los primeros tiempos del CP/M.
- Listado: Es una impresora de copia dura, bien de letras o de matriz de puntos.
- Unidades de disco: Estas pueden ser cualesquiera de las estándar de la industria, desde los disquetes de una sola cara, simple densidad y 8 pulgadas, a las de disco rígido con capacidad para varios cientos de megabytes.

Puntos de entrada del BIOS

Las primeras pocas instrucciones del BIOS son todas instrucciones de salto (JMP). Transfieren el control a las 17 subrutinas diferentes del BIOS. El CCP y el BDOS, cuando hacen una petición específica al BIOS, lo realizan transfiriendo el control a la instrucción JMP adecuada en la *tabla de saltos* o *vector de saltos* del BIOS. El vector de saltos del BIOS empieza siempre al principio de una página de 256 bytes, de forma que la dirección de la primera instrucción de salto es siempre de la forma xx00H, donde "xx" es la dirección de la página. Las posiciones 0000H a 0002H tienen una instrucción de salto a la segunda entrada del vector de saltos del BIOS —de forma que se pueda encontrar siempre la dirección de página del vector de saltos al mirar en el lugar 0002H.

La figura 6-1 muestra los contenidos del vector de saltos del BIOS junto con la dirección relativa de la página correspondiente a cada salto. Las etiquetas utilizadas en las instrucciones de salto se han adoptado por convenio.

Las secciones siguientes describen las funciones de cada una de las subrutinas principales del BIOS. Consúltese también el manual de Digital Research *Guía de modificación del CP/M 2.0* para la descripción de las rutinas del BIOS.

Funciones de arranque y carga

Existen dos funciones de arranque. El arranque en frío carga el sistema operativo completo cuando el sistema se conecta por primera vez o bien

```
xx00H    JMP   BOOT      ;Arranque en frío (primera vez)
xx03H    JMP   WBOOT     ;Arranque caliente
xx06H    JMP   CONST     ;Estado de entrada de la consola
xx09H    JMP   CONIN     ;Entrada de consola
xx0CH    JMP   CONOUT    ;Salida de consola
xx0FH    JMP   LIST      ;Salida de impresora
xx12H    JMP   PUNCH     ;Salida perforadora
xx15H    JMP   READER    ;Entrada lectora
xx18H    JMP   HOME      ;Situación de cabezas de disco en pista 0
xx1BH    JMP   SELDSK    ;Selección disco lógico
xx1EH    JMP   SETTRK    ;Fijación número de pista
xx21H    JMP   SETSEC    ;Fijación número de sector
xx24H    JMP   SETDMA    ;Fijación dirección de DMA
xx27H    JMP   READ      ;Lectura de sector (128 bytes)
xx2AH    JMP   WRITE     ;Escritura de sector (128 bytes)
xx2DH    JMP   LISTST    ;Estado dispositivo de listado de salida
xx30H    JMP   SECTRAN   ;Traducción de sector
```

Figura 6-1. Disposición del vector de saltos estándar del BIOS.

cuando se pone a cero. El arranque caliente recarga el CCP cuando un programa salta a la posición 0000H.

BOOT: Arranque "en frío"

La instrucción de salto BOOT es la primera instrucción ejecutada en el CP/M. La secuencia de arranque debe transferir el control al punto de entrada BOOT para poner a punto el CP/M. En general, una PROM recibe el control tanto cuando se aplica la energía por primera vez como después de haber pulsado el botón RESET de la computadora. Esta lee en el cargador del CP/M, en el primer sector de la unidad de disco físico elegido para ser el disco lógico A. Este programa de carga del CP/M lee la imagen binaria del CCP, el BDOS y el BIOS en la memoria en unas direcciones previamente determinadas. Entonces transfiere el control al punto de entrada BOOT del vector de saltos del BIOS.

Esta rutina BOOT debe inicializar todo el hardware de la computadora. Fija velocidad de comunicación para la consola física (si no se ha hecho ya durante la secuencia de arranque) y para las unidades de "lectura", de "perforación" y de listado, así como la unidad de disco. También deberá fijar la página base de memoria, de forma que haya un salto de la situación 0000H al punto de entrada de arranque caliente en el vector de saltos del BIOS (en xx03H) y un salto de la posición 0005H al punto de entrada del BDOS.

La mayor parte de las rutinas BOOT se manifiestan componiendo un mensaje en la consola, que indica la versión actual del CP/M y el hardware de la computadora que este BIOS puede soportar.

La rutina BOOT acaba transfiriendo el control al principio del CCP +6 bytes (el CCP tiene su propio vector de saltos pequeño al principio).

Exactamente antes de que la rutina BOOT salte al CCP, coloca el registro C a 0 para indicar que el disco lógico A será la unidad de disco por defecto. Esto es lo que provoca la "A>" a la indicación inicial del CCP.

El punto de entrada en curso del CCP se deriva de la dirección básica del BIOS. El CCP y el BDOS juntos requieren 1E00H bytes de programa, de forma que la primera instrucción del CCP se sitúa en la posición BIOS–1E00H.

WBOOT: Arranque "caliente"

A diferencia del punto de entrada de arranque en "frío", que se ejecuta sólo una vez, el WBOOT o rutina de arranque caliente se ejecuta cada vez que termina un programa, saltando a la posición 0000H, o siempre que se mecanografía un CONTROL-C en la consola como primer carácter de una línea de entrada.

La rutina WBOOT es responsable de la recarga del CCP en la memoria. Los programas utilizan a menudo toda la memoria hasta el punto de arranque del BDOS, reescribiendo sobre el CCP durante el proceso. La filosofía en que se basa es que mientras se está ejecutando un programa no se necesita el CCP, de forma que el programa puede utilizar la memoria ocupada previamente por éste. El CCP ocupa 800H (2.048) bytes de memoria —y esto es frecuentemente justo lo suficiente como para marcar la diferencia entre un programa que no puede funcionar y uno que sí puede.

Algunos programas que están contenidos en sí mismo y no requieren los servicios del BDOS superarán también el BDOS para obtener otros 1600H (5.632) bytes de memoria. Por consiguiente, para estar realmente a salvo, la rutina de WBOOT deberá leer tanto el CCP como en el BDOS. Para ello se necesita también fijar los dos JMP en la posición 0000H (al propio WBOOT) y en el lugar 0005H (al BDOS). La situación 0003H debería fijarse con el valor inicial del IOBYTE si éste está implementado en el BIOS.

En esta última acción, la rutina WBOOT fija el registro C para indicar el disco lógico seleccionado (C=0 para A, 1 para B, etc.). Entonces transfiere el control a la primera instrucción del CCP con el fin de ponerlo de nuevo en marcha. De nuevo, la dirección en curso se calcula basada en el conocimiento de que el CCP empieza 1E00H bytes más abajo en la memoria que la dirección básica del BIOS.

Funciones de entrada/salida de caracteres

Las funciones de entrada/salida de caracteres se enfrentan a los dispositivos lógicos: la consola, la "lectora", la "perforadora" y la impresora. Como

estos dispositivos lógicos pueden conectarse en la práctica por medio de software a uno o varios dispositivos físicos en entrada/salida de caracteres, muchos BIOS utilizan las características del IOBYTE del CP/M para asignar dispositivos lógicos a los físicos.

En este caso, cada una de las funciones del BIOS deberá controlar el campo de bits adecuado del IOBYTE (véanse figura 4-2 y tabla 4-1) para transferir el control al controlador del mecanismo físico correcto (programa que controla un mecanismo físico).

CONST: Modo de entrada por consola

CONST devuelve simplemente un indicador que muestra si existe un carácter de entrada desde el dispositivo de consola. Lo convencional es que A=00FFH si hay un carácter esperando para ser procesado, A=0 si no lo hay. Obsérvese que el signo cero no necesita colocarse para reflejar los contenidos de A en el registro; lo importante son los contenidos.

El CCP llama a CONST siempre que éste está en medio de una operación que puede ser interrumpida pulsando un carácter del teclado.

El BDOS llamará a CONST si un programa realiza una llamada a la función de lectura del estado de la consola (B$CONST, código 11, 0BH). También se le llama por medio de la rutina BIOS de entrada de consola, CONIN (descrita a continuación).

CONIN: Entrada de consola

CONIN lee el próximo carácter de la consola en el registro A y pone a cero el bit más significativo (paridad).

Normalmente, CONIN llamará la rutina CONST hasta que detecte A=0FFH. Sólo entonces introducirá el carácter de datos y enmascarará el bit de paridad.

CONIN es llamada por el CCP y por el BDOS cuando un programa ejecuta una función de lectura de byte de consola (B$CONIN, código 1).

CONOUT: Salida de consola

CONOUT envía el carácter (en ASCII) del registro C a la consola. El bit más significativo (paridad) de este carácter será siempre 0.

CONOUT deberá controlar, en primer lugar, que el dispositivo de consola esté dispuesto para recibir más datos, retrasándose si es necesario hasta haberlo comprobado y enviando sólo entonces el carácter al dispositivo.

CONOUT es llamado por el CCP y el BDOS cuando un programa ejecuta una función de escritura de byte en la consola (B$CONOUT, código 2).

LIST: Salida de listado

LIST es similar a CONOUT, excepto que envía el carácter del registro C al dispositivo de listado. También controla, en primer lugar, que esté preparado para recibir el carácter.

LIST es llamado por el CCP en respuesta a CONTROL-P para la repetición por impresora de la salida de consola y por el BDOS cuando un programa realiza una llamada de escritura de un byte por impresora o una llamada a la visualización de cadenas (B$LISTOUT y B$PRINTS, códigos 5 y 9).

PUNCH: Salida para "perforación"

PUNCH envía el carácter del registro C al dispositivo de "perforación". Como se ha dicho anteriormente, la "perforadora" no suele ser una perforadora real para cinta de papel. En la mayoría de los BIOS, el punto de entrada PUNCH o bien vuelve inmediatamente, con lo que es efectivamente una rutina anulada, o bien envía el carácter a un mecanismo de comunicaciones, tal como un "modem", de la computadora.

PUNCH debe controlar que el mecanismo de "perforación" se encuentre realmente preparado para aceptar otro carácter de salida y si no lo está habrá de esperar.

La documentación de Digital Research establece que el carácter que se ha de enviar tendrá siempre su bit más significativo puesto a cero. Esto no es cierto. El BDOS transfiere simplemente el control al punto de entrada PUNCH del BIOS; la fijación del bit más significativo será determinada por el programa que realiza la petición de función del BDOS (B$PUNOUT, código 4). Esto es importante porque el requerimiento de un cero impediría la posibilidad de enviar datos puramente binarios por medio de la función PUNCH del BIOS.

READER: Entrada de "lectora"

Como en el punto de entrada PUNCH, el punto de entrada READER rara vez conecta con una lectora real de tira de papel.

La función READER deberá devolver el próximo carácter desde el dispositivo de lectura al registro A, esperando, si es necesario, hasta que haya un carácter.

La documentación de Digital Research vuelve a decir que el bit más significativo del registro A deberá ser 0, pero este no es el caso si se desea recibir información puramente binaria a través de esta función.

READER es llamada siempre que un programa realiza una petición de función de lectura de byte en la "lectora" (B$READIN, código 3).

Funciones del disco

Todas las funciones del disco que siguen se habían diseñado originalmente para operar en sectores de 128 bytes utilizados en disquetes de una sola cara, simple densidad y 8 pulgadas, que fueron industrialmente estándar en su momento. Ahora que CP/M funciona con muchos tipos distintos de discos, algunas de las funciones BIOS del disco parecen extrañas porque la mayoría de los controladores de disco nuevos utilizan tamaños de sector distintos de 128 bytes.

Para manejar tamaños de sector mayores, el BIOS tiene algunos programas adicionales que hacen que el BDOS responda como si estuviera manejando aún sectores de 128 bytes. Este programa se denomina programa de *bloqueo/desbloqueo*. Como su nombre indica, acumula varios "sectores" de 128 bytes y escribe solamente en el disco cuando se ha completado un sector *físico*. Cuando lee, lo hace en un sector físico y entonces lo desbloquea, devolviendo algunos "sectores" de 128 bytes al BDOS.

Para hacer todo esto, el programa de bloqueo/desbloqueo utiliza un área especial de memoria intermedia del mismo tamaño que los sectores físicos del disco. Esto se conoce como memoria-intermedia-grande o HSTBUF. Los sectores físicos se leen en esta memoria y se escriben desde ella en el disco.

Para optimizar esta rutina de bloqueo/desbloqueo, el BIOS lo ha codificado para reducir el número de veces que se realiza una lectura o escritura de disco real. Un efecto secundario es que en cualquier momento dado se pueden almacenar algunos "sectores" de 128 bytes en el HSTBUF, esperando para ser escritos en el disco cuando el HSTBUF se llena. Esto complica a veces la lógica de las funciones BIOS del disco. No se puede seleccionar simplemente una nueva unidad de disco, por ejemplo, cuando el HSTBUF contiene datos destinados a otro. Esta complicación se verá en el BIOS únicamente en forma de aumento de operaciones lógicas; las funciones BIOS del disco raramente disparan operaciones físicas inmediatas. Es más fácil entender estas funciones BIOS si se considera que realizan *peticiones* —y que estas peticiones se satisfacen sólo cuando tiene sentido el hacerlo, teniendo en cuenta la lógica de bloqueo/desbloqueo.

HOME: Envío de disco al principio

HOME coloca la pista y el sector a 0.

SELDSK: Selección del disco

SELDSK no realiza lo que su nombre implica. No puede (ni debe) seleccionar físicamente un disco lógico. En lugar de ello, envía un indicador al par de registros HL de la cabecera de parámetros del disco para el disco lógico especificado en el registro C. C=0 para la unidad A, 1 para B, etc. SELDSK almacena también este código para que el disco requerido sea utilizado más tarde en las funciones READ y WRITE.

Si el código de disco lógico del registro C se refiere a un disco no existente o a uno para el cual no existe cabecera de parámetros de disco, entonces SELDSK deberá volver con HL puesto a 0000H. Entonces el BDOS enviará un mensaje en esta forma:

```
"BDOS Err on X: Select"
```

Obsérvese que SELDSK no sólo no selecciona el disco, sino que tampoco indica si el disco solicitado está físicamente presente o no —sino simplemente si existen tablas de discos presentes para el disco o no.

SELDSK es llamado por el BDOS bien durante las operaciones de fichero de disco o por un programa que lanza una petición de selección de disco (B$SELDSK, código 14).

SETTRK: Fijación de pista

SETTRK guarda la pista de disco que está en el par de registros BC cuando SETTRK obtiene el control. Obsérvese que éste es un número de pista absoluto; es decir, el número de pistas reservadas antes del directorio de ficheros se ha añadido al número de pistas correspondientes al principio del disco lógico.

El número de la pista requerida se utilizará en la próxima función READ o WRITE del BIOS (descritas posteriormente en este capítulo).

SETTRK es llamado por el BDOS cuando éste necesita leer o escribir un sector de 128 bytes. Los números de pista legítimos van de 0 a 0FFFFH (65.535).

SETSEC: Fijación de sector

SETSEC es similar al SETTRK en que almacena el número de sector requerido para uso posterior en las funciones de lectura o escritura (READ

o WRITE) del BIOS. El número de sector requerido es enviado por SETSEC al registro A; los valores legítimos van de 0 a 0FFH (255).

El número de sector es un número de sector lógico. No toma en cuenta ningún entrelazamiento de sectores que pueda utilizarse para mejorar el rendimiento del disco.

SETSEC es llamado por el BDOS cuando necesita leer o escribir un sector de 128 bytes.

SETDMA: Fijación de dirección DMA

SETDMA guarda la dirección en el par de registros BC en la dirección requerida de DMA. La próxima función READ o WRITE del BIOS utilizará la dirección DMA como apuntador a la memoria intermedia de un sector de 128 bytes en la que se leen o desde la cual se escribirán los datos.

La dirección por defecto de DMA es 0080H. SETDMA es llamada por el BDOS cuando necesita leer o escribir un sector de 128 bytes.

READ: Lectura de un sector

READ lee en un sector de 128 bytes con tal de que se hayan hecho llamadas previas del BIOS a

SELDSK — "selección" del disco.

SETDMA— fijación de la dirección del DMA.

SETTRK — fijación del número de pista.

SETSEC — fijación del número de sector.

Debido al programa de bloqueo/desbloqueo del BIOS, existen frecuentes ocasiones en las que el sector requerido esté ya en la memoria-intermedia-grande (HSTBUF), de forma que una lectura física del disco no es necesaria. Todo lo que se requiere entonces es que el BIOS transporte los 128 bytes adecuados desde el HSTBUF a la memoria intermedia señalada por la dirección de DMA.

Sólo durante la función READ se comunicará normalmente el BIOS con el controlador de disco físico, seleccionándolo e intentando leer la pista y el sector solicitados. Durante este proceso, la función READ deberá manejar también los errores de hardware que se presenten, intentando de nuevo una operación si ocurre un error "suave" o recuperable.

La función READ deberá volver con el registro A puesto a 00H si la operación de lectura se ha completado satisfactoriamente. Si la función

READ vuelve con el registro A puesto a 01H, el BDOS compondrá un mensaje de error en la forma:

```
BDOS Err on X: Bad Sector
```

En estas circunstancias existen sólo dos soluciones: se puede introducir un CARRIAGE RETURN, ignorar el hecho de que existía un error e intentar utilizar los datos de la memoria intermedia de DMA sensatamente. O se puede mecanografiar un CONTROL-C para abortar la operación, realizar un arranque y devolver el control al CCP.

Como puede verse, el manejo de errores del CP/M no ayuda demasiado, por lo que la mayoría de los autores de BIOS añaden procedimientos más sofisticados para anular errores precisamente en el controlador de disco. Esto puede incluir interacciones con la consola de forma que se pueda realizar un esfuerzo más determinado para corregir errores o, al menos, proporcionar mayor información acerca de dónde está el error. Este procesamiento de los errores se trata en el capítulo 9.

Si se trabaja con un sistema de disco duro, el controlador de disco debe tratar también el manejo de sectores estropeados. No se puede simplemente reemplazar una unidad de disco rígido si uno o dos de los sectores se hacen ilegibles. Este manejo de los sectores estropeados requiere normalmente que se ponga antes de usarlo un directorio de sectores "disponibles" en el disco rígido para almacenar datos. Entonces, cuando se encuentra que un sector está dañado, uno de los sectores "disponibles" se pone en lugar de éste. Esto se trata también en el capítulo 9.

WRITE: Escritura de sector

WRITE es similar a READ, pero con la diferencia obvia de que los datos se transfieren desde la memoria intermedia de DMA al sector específico de 128 bytes. Al igual que READ, esta función requiere que las llamadas a función que siguen hayan sido ya realizadas:

SELDSK — "selección" del disco.

SETDMA— fijación de la dirección de DMA.

SETTRK — fijación del número de pista.

SETSEC — fijación del número de sector.

Nuevamente, es únicamente la rutina WRITE la que el controlador pondrá en marcha para hablar directamente con el hardware físico, seleccionando la unidad de disco, la pista y el sector y transfiriendo los datos al disco.

Con el programa de bloqueo/desbloqueo, el BDOS optimiza el número

de escrituras de disco que se necesitan indicando en el registro C el tipo de escritura de disco que se ha de realizar:

0=escritura de sector normal.

1=escritura en el sector de directorio de ficheros.

2=escritura en el sector de bloques no usados previamente.

El tipo 0 se realiza siempre que el BDOS escribe en un sector de datos de un bloque ya utilizado. En estas circunstancias, el controlador de disco deberá leer previamente el sector-grande adecuado porque puede haberse almacenado previamente información en él.

El tipo 1 sucede siempre que el BDOS escribe en un sector del directorio de ficheros —en este caso, el BIOS no debe diferir la escritura del sector al disco, porque la información es demasiado valiosa para conservarla en memoria hasta que el HSTBUF esté lleno—. Cuando más larga sea la información que reside en el HSTBUF, mayor será la oportunidad de que exista un fallo de energía o caída de tensión haciendo inaccesibles los datos ya escritos físicamente en el fichero en el disco porque el directorio no ha sido actualizado.

El tipo 2 ocurre siempre que el BDOS necesita escribir en el primer sector de un bloque no usado previamente. No usado, en este contexto, incluye un bloque que se ha hecho disponible como consecuencia del borrado de un fichero. En este caso no hay necesidad de que el controlador de disco haga la prelectura de un sector de tamaño grande en el HSTBUF, ya que no existen datos de valor en el sector físico.

Igual que con la rutina READ, la función WRITE vuelve con A puesto a 00H si la operación se ha completado satisfactoriamente. Si la función WRITE vuelve con A puesto a 01H, entonces el BDOS compondrá *el mismo* mensaje que para READ:

```
BDOS Err on X: Bad Sector
```

Ahora puede verse por qué la mayor parte de los autores de BIOS añaden rutinas más extensas de recuperación de errores y de interacción con el usuario a sus controladores de disco.

Para sistemas de discos rígidos, algunos controladores de disco se escriben de tal forma que "salvan" automáticamente un sector que falla, escribiendo los datos en uno de los sectores disponibles del disco.

LISTST: Modo de listado

Como puede deducirse de su posición en la lista de las funciones del BIOS, la función LISTST es un añadido. Se agregó cuando el CP/M se varió de la versión 1.4 a la versión 2.0.

Esta función devuelve el modo en curso del dispositivo de listado, utilizando si es necesario el IOBYTE para seleccionar el dispositivo físico correcto. Pone el registro A a 0FFH si el dispositivo de listado puede aceptar otro carácter para salida o a 00H si no está preparado.

La documentación de Digital Research establece que esta función se utiliza por el programa de utilidad DESPOOL (que permite imprimir un fichero "simultáneamente" con otras operaciones) para mejorar la respuesta de la consola durante su operación, y que es siempre aceptable para la rutina devolver 00H si se escoge no implementarla completamente.

Desgraciadamente, esta aseveración es falsa. Muchos otros programas utilizan la función LISTST para "comprobar" el dispositivo de listado a fin de asegurar que esté dispuesto, y si no está dispuesto después de un tiempo determinado previamente, se envía un mensaje a la consola indicando que la impresora no está preparada. Si se hace una llamada a las funciones de salida de listado del BDOS, escritura impresión del byte en impresora (códigos 5 y 9) y la impresora no está preparada, entonces el CP/M esperará eternamente —y el programa habrá perdido el control, de forma que no puede detectar que ha ocurrido un problema—. Si el LISTST vuelve siempre a 00H, entonces la impresora aparecerá siempre como no preparada. No sólo resulta inservible la función del LISTST, sino que también puede causar un aluvión de mensajes de error falsos "Impresora no preparada" que aparecerán en la consola.

SECTRAN: Traducción de sector

SECTRAN, dado un número de sector lógico, coloca el número de sector físico correcto en la tabla de traducción de sectores para el controlador de disco lógico previamente seleccionado (por medio del SELDSK).

Obsérvese que tanto los números de sector lógico como los de sector físico son sectores de 128 bytes, de forma que si se está trabajando con un sistema de disco rígido, no es demasiado eficaz imponer un entrelazado de sectores a nivel del sector de 128 bytes. Es mejor imponer el entrelazado de sectores justamente dentro del controlador de disco rígido, si se realiza; en general, los discos rígidos giran tan rápidamente que el CP/M sencillamente no puede beneficiarse del entrelazado de sectores.

El BDOS maneja el número de sector lógico en el par de registros BC, con la dirección de la tabla de traducción del sector en el par de registros DE. SECTRAN deberá devolver el número de sector físico en HL.

Si SECTRAN es una rutina nula, deberá transferir los contenidos de BC a HL y volver.

Llamada a las funciones del BIOS directamente

Como regla general, no se deben hacer llamadas directas al BIOS. Al hacerlas, los programas se vuelven menos transportables de un sistema CP/M a otro. Impide la posibilidad de que estos programas funcionen con un MP/M que tenga una forma diferente de BIOS, denominada sistema de entrada/salida extendido, o XIOS.

Existen uno o dos problemas, sin embargo, que sólo pueden resolverse haciendo llamadas directas al BIOS. Esto ocurre en los programas de utilidad que, por ejemplo, necesitan realizar un acceso directo al directorio de ficheros del CP/M, o necesitan acceder a algunas instrucciones "particulares" de salto que ya han sido añadidas al vector de saltos estándar del BIOS.

Si se necesita realmente acceso directo al BIOS, la figura 6-2 muestra un ejemplo de subrutina que lo hace. Requiere que el registro A contenga un código de función del BIOS que indique el desplazamiento, en el vector de saltos, de la instrucción de salto a la cual se ha de pasar el control.

```
                ;       Equivalencias para utilizar con subrutina BIOS
                ;
0003 =          WBOOT   EQU     03H     ;Arranque caliente
0006 =          CONST   EQU     06H     ;Estado de la consola
0009 =          CONIN   EQU     09H     ;Entrada de consola
000C =          CONOUT  EQU     0CH     ;Salida de consola
000F =          LIST    EQU     0FH     ;Salida a dispositivo de impresión
0012 =          PUNCH   EQU     12H     ;Salida a dispositivo de perforación
0015 =          READER  EQU     15H     ;Entrada de lectora
0018 =          HOME    EQU     18H     ;Puesta en pista 0 del disco seleccionado
001B =          SELDSK  EQU     1BH     ;Selección de disco
001E =          SETTRK  EQU     1EH     ;Fijación de pista
0021 =          SETSEC  EQU     21H     ;Fijación sector
0024 =          SETDMA  EQU     24H     ;Fijación dirección DMA
0027 =          READ    EQU     27H     ;Lectura de sector de 128 bytes
002A =          WRITE   EQU     2AH     ;Escritura de sector de 128 bytes
002D =          LISTST  EQU     2DH     ;Retorno estado de impresora
0030 =          SECTRAN EQU     30H     ;Traducción de sectores
                ;
                                        ;Añadir aquí programas "privados" BIOS

                ;
                ;       BIOS
                ;       Esta rutina transfiere control a la entrada apropiada del vector
                ;       de saltos del BIOS, basado en un código numérico que se maneja
                ;       para ello en el registro L.
                ;
                ;       Parámetros de entrada
                ;
                ;       L = número de código (en realidad es la dirección
                ;       relativa-a-la-página de la instrucción JMP correcta
                ;       en el vector de saltos.
                ;       Todos los demás registros son protegidos y llevados sin manipular
                ;       fuera de la rutina del BIOS.
                ;
                ;       Parámetros de salida
                ;
```

Figura 6-2. Equivalencias del BIOS.

```
                        ;       Esta rutina no llama a la rutina BIOS, por lo que cuando
                        ;       ésta RETorna lo hace directamente sobre aquella
                        ;       que la ha llamado.
                        ;
                        ;       Secuencia de llamada
                        ;
                        ;               MVI     L,Code$Number
                        ;               CALL    BIOS
                        ;
                        BIOS:
0000 F5                         PUSH    PSW     ;Almacenar registro-de-usuario A
0001 3A0200                     LDA     0002H   ;Obtención página del vector de saltos del BIOS
                                                ;  a partir de JMP-arranque-caliente
0004 67                         MOV     H,A     ;HL -> entrada al vector de saltos del BIOS
0005 F1                         POP     PSW     ;Recuperación del registro-de-usuario A
0006 E9                         PCHL            ;Transferencia del control a la rutina BIOS
```

Figura 6-2. (Continuación.)

Números de línea	Componente funcional o rutina
0072-0116	Vector de saltos del BIOS
0120-0270	Programa de inicialización
0275-0286	Visualización de mensaje
0289-0310	Cargar CP/M
0333-0364	CONST - Estado de la consola
0369-0393	CONIN - Entrada consola
0397-0410	CONOUT - Salida consola
0414-0451	LISTST - Estado de impresión
0456-0471	LIST - Salida de impresión
0476-0492	PUNCH - Salida de perforación
0496-0511	READER - Entrada de lectora
0516-0536	IOBYTE - Selección de dispositivo
0540-0584	Tablas de control de dispositivo
0589-0744	Controladores de bajo nivel para consola, impresora, etc.
0769-0824	Tablas cabecera de parámetros de disco
0831-0878	Bloques de parámetros de disco
0881-0907	Otras áreas de datos de disco
0910-0955	SELDSK - Selección de disco
0958-0964	SETTRK - Fijación de pista
0967-0973	SETSEC - Fijación de sector
0978-0984	SETDMA - Fijación dirección de DMA
0987-1025	Tablas de desplazamiento de sectores
1028-1037	SECTRAN - Traducción de sectores lógicos a físicos
1041-1056	HOME - Vuelta a la pista 0
1059-1154	Areas de datos del algoritmo de desbloqueo
1157-1183	READ - Lectura de sector de 128 bytes
1185-1204	WRITE - Escritura de sector de 128 bytes
1206-1378	Algoritmo de desbloqueo
1381-1432	Transferencia a la memoria intermedia
1435-1478	Subrutinas de desbloqueo
1481-1590	Lectura/escritura en disco flexible físico de 8''
1595-1681	Lectura/escritura en disco flexible físico de 5 1/4''
1685-1764	WBOOT - Arranque caliente

Figura 6-3. Indice funcional de la figura 6-4.

Ejemplo de BIOS

El resto de este capítulo se dedica a un ejemplo de listado del BIOS. Este BIOS, trabajando prácticamente, muestra la escritura general y la interfase con las subrutinas del BIOS.

A diferencia de la mayoría de los BIOS, éste se ha escrito específicamente para ser entendido con facilidad. Los nombres variables son desacostumbradamente largos y descriptivos y cada bloque de programa lleva un comentario para situarlo en el contexto.

Cada línea de código fuente ha sido numerada secuencialmente (una opción usada poco frecuentemente que permite el ensamblador ASM de Digital Research). La figura 6-3 contiene un índice funcional del BIOS como un conjunto, de forma que se puedan encontrar funciones particulares en el listado de la figura 6-4 mediante el número de línea.

```
0001 <-- Número de linea ;    Figura 6-4.
0002                     ;
0003                     ;*********************************************************
0004                     ;*                                                       *
0005                     ;*              Listado del BIOS                         *
0006                     ;*                                                       *
0007                     ;*********************************************************
0008                     ;
0009                     ;
0010  3030 =             VERSION             EQU     '00'     ;Equivalencias utilizadas en el mensaje
0011  3730 =             MONTH               EQU     '07'     ; de señal de "funcionando"
0012  3531 =             DAY                 EQU     '15'
0013  3238 =             YEAR                EQU     '82'
0014                     ;
0015                     ;*************************************************************************
0016                     ;*                                                                       *
0017                     ;*  Este BIOS es para un sistema de cálculo con la siguiente             *
0018                     ;*  configuración hardware:                                              *
0019                     ;*                                                                       *
0020                     ;*      - CPU 8080.                                                      *
0021                     ;*      - 64K bytes de RAM.                                              *
0022                     ;*      - Controlador de teclado/CRT que transfiere datos como si fuera un *
0023                     ;*        puerto serie [pero sin necesitar un generador de impulsos      *
0024                     ;*        para control de velocidad de transmisión (baud rate generator) o *
0025                     ;*        programación de la USART].                                     *
0026                     ;*      - Un puerto serie, usado por la impresora o por los dispositivos *
0027                     ;*        de listado, lectura o cinta perforada. Este chip es un         *
0028                     ;*        INTEL 8251A con un generador de control-de-velocidad 8253.     *
0029                     ;*      - Dos unidades de discos flexibles de 5 1/4'', doble cara, doble *
0030                     ;*        densidad. Estos dispositivos utilizan sectores de 512 bytes    *
0031                     ;*        que son utilizados como discos lógicos A: y B:.                *
0032                     ;*      - Dos unidades de disco estándar de 8'' (sectores de 128 bytes)  *
0033                     ;*        que se usarán como discos lógicos C: y D:.                     *
0034                     ;*                                                                       *
0035                     ;*        Se utilizan dos controladores de disco inteligentes,           *
0036                     ;*        uno para cada tipo de disquete. Estos controladores acceden    *
0037                     ;*        directamente a la memoria para leer los detalles               *
0038                     ;*        de las operaciones que deben realizar y para leer y escribir   *
0039                     ;*        datos desde o en los disquetes.                                *
0040                     ;*                                                                       *
0041                     ;*************************************************************************
0042
0043                     ;
0044                     ;  Las equivalencias para definir el tamaño de memoria, la dirección base
0045                     ;  y la longitud de los componentes del sistema son:
```

Figura 6-4. Listado del BIOS simple.

```
0046                    ;
0047  0040 =            Memory$Size         EQU     64      ;Número de Kbytes de RAM
0048                    ;
0049                    ;  La longitud del BIOS puede determinarse por inspección.
0050                    ;  Cambiando el primer carácter de la línea ORG BIOS$Entry, más abajo,
0051                    ;  por un punto y coma se pondrá ésta como línea de comentario. (Esto
0052                    ;  hará que el ensamblador comience el BIOS en la posición 0.) Ensamblar
0053                    ;  el BIOS y marcar en la posición próxima a 100H la dirección que se
0054                    ;  visualiza en la consola al final del ensamblaje.
0055                    ;
0056  0900 =            BIOS$Length         EQU     0900H
0057                    ;
0058  0800 =            CCP$Length EQU      0800H   ;Constante
0059  0E00 =            BDOS$Length         EQU     0E00H   ;Constante
0060                    ;
0061  0008 =            Overall$Length      EQU     ((CCP$Length + BDOS$Length + BIOS$Length) / 1024) + 1
0062                    ;
0063  E000 =            CCP$Entry  EQU      (Memory$Size - Overall$Length) * 1024
0064  E806 =            BDOS$Entry EQU      CCP$Entry + CCP$Length + 6
0065  F600 =            BIOS$Entry EQU      CCP$Entry + CCP$Length + BDOS$Length
0066                    ;
0067                    ;
0068                    ;
0069
0070  F600                ORG     BIOS$Entry      ;Código de ensamblaje en la dirección del BIOS
0071                    ;
0072                    ;  Vector de saltos del BIOS.
0073                    ;  El control se transferirá desde el CCP o el BDOS al punto
0074                    ;  de entrada adecuado, ambos calculan la dirección relativa del vector
0075                    ;  de saltos del BIOS para localizarlo.
0076                    ;  Los programas transitorios pueden hacer también llamadas directas
0077                    ;  al BIOS transfiriendo control a la posición xx00H, donde xx es el
0078                    ;  valor situado en la posición 0002H.
0079                    ;
0080  F600 C3F9F6       JMP     BOOT    ;Arranque en frío -- entrado desde el cargador de CP/M
0081                  Warm$Boot$Entry:  ;  Etiquetado de forma que el programa de inicialización pueda situar
0082                                    ;  la dirección de entrada de arranque caliente en la posición
0083                                    ;  0001H y 0002H de la página base
0084  F603 C329FE       JMP     WBOOT   ;Arranque caliente -- entrado saltando a la posición 0000H
0085                                    ;  Recarga el CCP que puede haber sido reescrito
0086                                    ;  destructivamente por el programa anterior en el área
0087                                    ;  de programas transitorios
0088  F606 C362F8       JMP     CONST   ;Estado de consola -- devuelve A=0FFH si existe un carácter esperando
0089                                    ;  en la consola
0090  F609 C378F8       JMP     CONIN   ;Entrada de consola -- devuelve en A el siguiente carácter
0091                                    ;  del teclado de la consola
0092  F60C C386F8       JMP     CONOUT  ;Salida de consola -- lanza al dispositivo
0093                                    ;  de consola el carácter que está en C
0094  F60F C3ACF8       JMP     LIST    ;Salida de impresora -- lanza al dispositivo de impresión el carácter
0095                                    ;  que está en C
0096  F612 C3BCF8       JMP     PUNCH   ;Salida de perforadora -- lanza al dispositivo lógico de perforación
0097                                    ;  el carácter que está en C
0098  F615 C3CDF8       JMP     READER  ;Entrada de lectora -- sitúa en A el siguiente carácter de entrada
0099                                    ;  del dispositivo de lectura
0100  F618 C3D3FB       JMP     HOME    ;Sitúa en la pista 0 al disco seleccionado en curso
0101  F61B C32BFB       JMP     SELDSK  ;Selecciona la unidad de disco especificada en C y devuelve la
0102                                    ;  dirección de los parámetros cabecera del disco
0103  F61E C358FB       JMP     SETTRK  ;Fija, a partir del par de registro BC, la pista del disco para la
0104                                    ;  próxima operación de lectura o escritura
0105  F621 C35EFB       JMP     SETSEC  ;Fija, a partir del registro A, el sector del disco para la próxima
0106                                    ;  operación de lectura o escritura
0107  F624 C365FB       JMP     SETDMA  ;Fija, a partir del par de registro DE, la dirección de memoria de
0108                                    ;  acceso directo para la próxima op. de lectura o escritura
0109
0110  F627 C3FBFB       JMP     READ    ;Lee sobre la dirección de DMA la pista y sector del disco
0111                                    ;  seleccionado especificado previamente
0112  F62A C315FC       JMP     WRITE   ;Escribe en el disco seleccionado la pista y sector especificados en
0113                                    ;  el disco seleccionado desde la dirección de DMA
0114  F62D C394F8       JMP     LISTST  ;Devuelve A = 0FFH si el dispositivo de impresión puede aceptar
0115                                    ;  otro carácter de salida
0116  F630 C3CDFB       JMP     SECTRAN ;Traduce un sector lógico en uno físico
0117                    ;
0118                    ;
0119                    ;
0120                    ;  El código de inicialización de arranque en frío es necesario una sola vez.
```

Figura 6-4. (Continuación.)

```
0121                    ;   Puede ser reescrito destructivamente una vez que ha sido ejecutado.
0122                    ;   A pesar de ello, está "escondido" en la memoria intermedia principal del disco.
0123                    ;   Cuando se transfiere el control al punto de entrada de autocarga (BOOT),
0124                    ;   este programa se ejecuta, y sólo se escribirá sobre él destructivamente
0125                    ;   datos del disco una vez que el proceso de inicialización se ha completado.
0126                    ;
0127                    ;   Para guardar programas en la memoria intermedia, ésta se declara primero normalmente.
0128                    ;   En este momento se anota el valor del contador de posiciones que sigue
0129                    ;   a la memoria intermedia. Entonces, utilizando una instrucción ORG (origen),
0130                    ;   el contador de posiciones es "vuelto atrás" al comienzo de la memoria y el programa
0131                    ;   de inicialización escrito normalmente.
0132                    ;   Al final de este programa se utiliza otra instrucción ORG
0133                    ;   para fijar el contador de posiciones donde estaba antes
0134                    ;   de que la memoria se declarase.
0135                    ;
0136                    ;
0137  0200 =            Physical$Sector$Size        EQU     512     ;Este es el tamaño real de sector
0138                                                        ;para disquetes de 5 1/4"
0139                                                        ;Los disquetes de 8" utilizan sectores de 128 bytes
0140                                                        ;Se declara la memoria de disco físico
0141                                                        ;para disquetes de 5 1/4"
0142  F633              Disk$buffer:        DS      Physical$Sector$Size
0143                    ;
0144                                                        ;Guardar contador de posiciones
0145  F833 =            After$Disk$Buffer   EQU     $       ;$ = Valor en curso del contador de posiciones
0146                    ;
0147  F633                                  ORG     Disk$Buffer     ;Situar atrás el contador de posiciones
0148                    ;
0149                    Initialize$Stream: ;Este flujo de datos se utiliza
0150                                       ;para inicializar la subrutina. Tiene
0151                                       ;el siguiente formato:
0152                                       ;
0153                                       ;       DB      Número de puerto a inicializar
0154                                       ;       DB      Número de bytes a lanzar a salida
0155                                       ;       DB      xx,xx,xx,xx datos a lanzar a salida
0156                                       ;       :
0157                                       ;       :
0158                                       ;       DB      Número de puerto de terminal 00H
0159                                       ;
0160                                       ;Nota : En esta máquina el puerto de consola no precisa
0161                                       ;       ser inicializado. Lo hace casi totalmente
0162                                       ;       el código cargador en PROM.
0163                                       ;
0164                                                ;Inicialización de la USART 8251A utilizada por los
0165                                                ;  dispositivos de comunicación y de listado
0166  F633 ED               DB      Communication$Status$Port       ;Número de puerto
0167  F634 06               DB      6                               ;Número de bytes
0168  F635 00               DB      0               ;Preparación del chip para ser programado
0169  F636 00               DB      0               ;  enviando datos ficticios
0170  F637 00               DB      0
0171  F638 42               DB      0100$0010B      ;Puesta a cero y armado del terminal de datos
0172  F639 6E               DB      01$10$11$10B    ;1 bit de parada, paridad-no, 8 bits por carácter y factor
0173                                                ;  de división de 16 como control-de-velocidad (baud rate)
0174  F63A 25               DB      0010$0101B      ;Elevar petición para mandar y dispuesto
0175                                                ;  para transmitir y recibir
0176                    ;
0177                                                ;Inicializar el temporizador programable 8352
0178                                                ;  usado para generar el control de
0179                                                ;  velocidad de la USART 8251A
0180  F63B DF               DB      Communication$Baud$Mode         ;Número de puerto
0181  F63C 01               DB      1                               ;Número de bytes
0182  F63D B6               DB      10$11$011$0B    ;Selección contador 2, cargar primer byte de LS
0183                                                ;  Contador binario módulo 3 (para control-de-velocidad)
0184                    ;
0185  F63E DE               DB      Communication$Baud$Rate         ;Número de puerto
0186  F63F 02               DB      2                               ;Número de bytes
0187  F640 3800             DW      0038H           ;1200 baudios (basado en la división por 16
0188                                                ;  seleccionada en la USART 8251A)
0189                    ;
0190  F642 00               DB      0               ;Número de puerto de 0 terminaciones
0191                    ;
0192                    ;
0193                    ;   Equivalencia para mensajes de "funcionando"
0194                    ;
0195  000D =            CR EQU      0DH             ;Retorno de carro
```

Figura 6-4. (Continuación.)

```
0196  000A =          LF EQU     0AH              ;Salto de linea
0197                  ;
0198                  Signon$Message:             ;Mensaje de "funcionando" principal
0199  F643 43502F4D20    DB      'CP/M 2.2.'
0200  F64C 3030           DW      VERSION          ;Número de versión en curso
0201  F64E 20             DB      ' '
0202  F64F 3037           DW      MONTH            ;Fecha del dia
0203  F651 2F             DB      '/'
0204  F652 3135           DW      DAY
0205  F654 2F             DB      '/'
0206  F655 3832           DW      YEAR
0207  F657 0D0A0A         DB      CR,LF,LF
0208  F65A 53696D706C     DB      'Simple BIOS',CR,LF,LF
0209  F668 4469736B20     DB      'Disk configuration :',CR,LF,LF
0210  F67F 2020202020     DB      '      A: 0.35 Mbyte 5" Floppy',CR,LF
0211  F69D 2020202020     DB      '      B: 0.35 Mbyte 5" Floppy',CR,LF,LF
0212  F6BC 2020202020     DB      '      C: 0.24 Mbyte 8" Floppy',CR,LF
0213  F6DA 2020202020     DB      '      D: 0.24 Mbyte 8" Floppy',CR,LF
0214                  ;
0215  F6F8 00             DB      0
0216                  ;
0217  0004 =          Default$Disk        EQU     0004H   ;Disco por defecto en página base
0218                  ;
0219                  BOOT:       ; Entrada directa desde el vector de saltos del BIOS.
0220                              ; El control será transferido directamente aquí
0221                              ;   por el cargador de CP/M
0222                              ; El estado de inicialización del sistema de cálculo
0223                              ;   estará determinado por el cargador en PROM
0224                              ;   y por el cargador de CP/M activado
0225                              ;
0226                                                  ;Sistema de inicialización
0227                                                  ;Esta rutina utiliza la Initialize$Stream
0228                                                  ;  declarada anteriormente
0229  F6F9 F3             DI                        ;Desactivación de interrupciones para prevenir cualquier
0230                                                  ;  tipo de efectos durante la inicialización
0231  F6FA 2133F6         LXI     H,Initialize$Stream     ;HL -> flujo de datos
0232                  ;
0233                  Initialize$Loop:
0234  F6FD 7E             MOV     A,M               ;Obtención de número de puerto
0235  F6FE B7             ORA     A                 ;Si es 00H, inicialización completada
0236  F6FF CA13F7         JZ      Initialize$Complete
0237  F702 320AF7         STA     Initialize$Port ;Activación de la instrucción OUT
0238  F705 23             INX     H                 ;HL -> cuenta del número de bytes a lanzar a salida
0239  F706 4E             MOV     C,M               ;Obtención cuenta de bytes
0240                  ;
0241                  Initialize$Next$Byte:       ;HL -> siguiente byte de datos
0242  F707 23             INX     H                 ;Obtención siguiente byte de datos
0243  F708 7E             MOV     A,M               ;Salida a puerto correcto
0244  F709 D3             DB      OUT
0245                  Initialize$Port:
0246  F70A 00             DB      0                 ;<- Fijar anterior
0247  F70B 0D             DCR     C                 ;Descontar
0248  F70C C207F7         JNZ     Initialize$Next$Byte    ;Vuelta atrás si existen más bytes
0249  F70F 23             INX     H                 ;HL -> número de puerto siguiente
0250  F710 C3FDF6         JMP     Initialize$Loop ;Vuelta atrás para inicialización puerto siguiente
0251                  ;
0252                  Initialize$Complete:
0253                  ;
0254
0255  F713 3E01           MVI     A,00$00$00$01B          ;Fijación del IOBYTE para indicar que el terminal
0256  F715 320300         STA     IOBYTE                  ;  actúa como consola
0257
0258  F718 2143F6         LXI     H,Signon$Message        ;Visualización en la consola de mensaje de
0259  F71B CD33F8         CALL    Display$Message         ;  "funcionando"
0260                  ;
0261
0262  F71E AF             XRA     A                 ;Fijar a A: unidad de disco por defecto
0263  F71F 320400         STA     Default$Disk
0264  F722 FB             EI                        ;Activar interrupciones
0265                  ;
0266  F723 C340F8         JMP     Enter$CPM         ;Completar la inicialización y entrar
0267                                                ;  en CP/M yendo al procesador de comandos
0268                                                ;  de consola
0269                  ;
0270                  ;  Final del programa de inicialización de arranque en frio
0271                  ;
```

Figura 6-4. (Continuación.)

```
0272  F833                ORG     After$Disk$Buffer       ;Puesta a cero del contador de posiciones
0273                      ;
0274                      ;
0275                      Display$Message:  ;Visualizar el mensaje especificado en la consola
0276                                        ;En entrada, HL apunta al flujo de bytes a lanzar a la salida
0277                                        ;  Termina el mensaje un byte 00H
0278  F833 7E             MOV     A,M             ;Obtención siguiente byte de la memoria intermedia
0279  F834 B7             ORA     A               ;Comprobación de terminal
0280  F835 C8             RZ                      ;Sí, vuelta a la rutina que efectuó la llamada
0281  F836 4F             MOV     C,A             ;Preparación para lanzar a la salida
0282  F837 E5             PUSH    H               ;Almacenar apuntador a memoria intermedia
0283  F838 CD86F8         CALL    CONOUT          ;Ir a la rutina principal de salida de consola
0284  F83B E1             POP     H               ;Recuperación de apuntador a memoria intermedia
0285  F83C 23             INX     H               ;Traslado al siguiente byte de la memoria
0286  F83D C333F8         JMP     Display$Message ;Bucle hasta terminar salida de la memoria
0287                      ;
0288                      ;
0289                      Enter$CPM:  ;Esta rutina se entra a partir del arranque en frío o en caliente.
0290                                  ;  Activa las instrucciones JMP en la página base y fija la dirección
0291                                  ;  de entrada/salida de unidades de disco de alto nivel (conocida
0292                                  ;  también como dirección de DMA).
0293                      ;
0294  F840 3EC3           MVI     A,JMP           ;Obtención código máquina para JMP
0295  F842 320000         STA     0000H           ;Activación JMP en la posición 0000H
0296  F845 320500         STA     0005H           ;  y 0005H
0297                      ;
0298  F848 2103F6         LXI     H,Warm$Boot$Entry       ;Obtención de la dirección del vector BIOS
0299  F84B 220100         SHLD    0001H           ;Puesta de la dirección en la posición 0001H
0300
0301  F84E 2106E8         LXI     H,BDOS$Entry    ;Obtención punto de entrada al BDOS
0302  F851 220600         SHLD    6               ;Puesta de la dirección en la posición 0005H
0303                      ;
0304  F854 018000         LXI     B,80H           ;Fijación de dirección de ent./sal. de disco por defecto
0305  F857 CD65FB         CALL    SETDMA          ;Utilización rutina normal BIOS
0306                      ;
0307  F85A FB             EI                      ;Asegurar interrupciones activadas
0308  F85B 3A0400         LDA     Default$Disk    ;Transferencia disco en curso por defecto al
0309  F85E 4F             MOV     C,A             ;  procesador de órdenes de consola
0310  F85F C300E0         JMP     CCP$Entry       ;Transferencia al CCP
0311                      ;
0312                      ;
0313                      ;  Controladores de entrada/salida serie
0314                      ;
0315                      ;  Estos controladores examinan el IOBYTE en la posición 0003H
0316                      ;  que ha sido fijado por la rutina de arranque en frío.
0317                      ;  El IOBYTE puede ser modificado por la rutina de utilidad STAT,
0318                      ;  por llamadas al BDOS, o por un programa que sitúe un valor
0319                      ;  directamente en la posición 0003H.
0320                      ;
0321                      ;  Todas estas rutinas hacen uso de la subrutina Select$Routine,
0322                      ;  que toma los dos bits menos significativos del registro A
0323                      ;  y los utiliza para transferir el control a una de las rutinas
0324                      ;  cuya dirección sigue inmediatamente a la llamada a Select$Routine.
0325                      ;  Un segundo punto de entrada Select$Routine$21 utiliza
0326                      ;  los bits 2 y 1 para hacer la misma operación -- ésta guarda
0327                      ;  espacio para salvar una instrucción innecesaria.
0328                      ;
0329  0003 =              IOBYTE  EQU     0003H   ;Byte de redireccionamiento de entrada/salida
0330                      ;
0331                      ;
0332                      ;
0333                      CONST:              ;Obtención estado de consola
0334                                          ;Entrada directamente desde el vector JMP del BIOS
0335                                          ;  devuelve un parámetro que refleja la existencia
0336                                          ;  de datos a entrar desde la consola
0337                                          ;
0338                                          ;A = 00H (marca de cero activada) si no hay datos
0339                                          ;A = 0FFH (marca de cero a cero) si existen datos
0340                                          ;
0341                                          ;CONST es llamada por programas que realizan
0342                                          ;  comprobaciones periódicas para indagar si el operador
0343                                          ;  ha pulsado algunas teclas -- por ejemplo, para interrumpir
0344                                          ;  un programa en ejecución
0345
0346  F862 CD6AF8         CALL    Get$Console$Status      ;Retorno A = cero o distinto de cero
0347                                                      ;De acuerdo con el estado, convertir después
```

Figura 6-4. (Continuación.)

```
0348                                        ;  para seguir la convención de parámetros
0349  F865 B7          ORA     A            ;Activación de marcas para reflejar el estado
0350  F866 C8          RZ                   ;Si 0, no existen datos de entrada
0351  F867 3EFF        MVI     A,0FFH       ;En caso contrario, devolución de A = 0FFH para indicar
0352  F869 C9          RET                  ;  existencia de datos de entrada
0353                ;
0354                Get$Console$Status:
0355  F86A 3A0300      LDA     IOBYTE       ;Obtención de byte de redireccionamiento de entrada/salida
0356                                        ;Se selecciona la consola de acuerdo con
0357                                        ;  los bits 1 y 0 del IOBYTE
0358  F86D CDDCF8      CALL    Select$Routine ;Selección de la rutina apropiada
0359                                        ;Estas rutinas devuelven el control a la que efectuó
0360                                        ;  la llamada a Get$Console$Status
0361  F870 F6F8        DW      Teletype$In$Status      ;00 <- bits 1,0 del IOBYTE
0362  F872 FCF8        DW      Terminal$In$Status      ;01
0363  F874 02F9        DW      Communication$In$Status ;10
0364  F876 08F9        DW      Dummy$In$Status         ;11
0365                ;
0366                ;
0367                ;
0368                ;
0369                CONIN:              ;Obtención de carácter de entrada de consola
0370                                    ;Entrada directamente del vector JMP del BIOS;
0371                                    ;  devuelve en el registro A el siguiente carácter
0372                                    ;  de datos de la consola. El bit más significativo
0373                                    ;  del carácter-dato será 0 excepto cuando se haya
0374                                    ;  seleccionado la lectora (puerto de comunicaciones).
0375                                    ;  En este caso los ocho bits de datos completos
0376                                    ;  serán devueltos para permitir la recepción de datos
0377                                    ;  binarios
0378                                    ;
0379                                    ;Normalmente, esta rutina será llamada después
0380                                    ;  de que una llamada a CONST indique que está preparado
0381                                    ;  un carácter-dato, pero en cualquier caso el CCP o el BDOS
0382                                    ;  no pueden hacer nada hasta que ocurre una entrada
0383                                    ;  por la consola, entonces CONIN será llamada sin una llamada
0384                                    ;  anterior a CONST
0385                                    ;
0386  F878 3A0300      LDA     IOBYTE          ;Obtención de byte de redireccionamiento de entrada/salida
0387  F87B CDDCF8      CALL    Select$Routine  ;Selección de la rutina CONIN correcta
0388                                                ;Estas rutinas devuelven control directamente a la
0389                                                ;  rutina que llamó a CONIN
0390  F87E 20F9        DW      Teletype$Input          ;00 <- bits 1,0 del IOBYTE
0391  F880 26F9        DW      Terminal$Input          ;01
0392  F882 2FF9        DW      Communication$Input     ;10
0393  F884 35F9        DW      Dummy$Input             ;11
0394                ;
0395                ;
0396                ;
0397                CONOUT:             ;Salida de consola
0398                                    ;Entrada directamente desde el vector JMP del BIOS;
0399                                    ;  lanza a salida el carácter situado en el registro C
0400                                    ;  sobre el dispositivo apropiado de acuerdo con los
0401                                    ;  bytes 1 y 0 del IOBYTE
0402                                    ;
0403  F886 3A0300      LDA     IOBYTE          ;Obtención del byte de redireccionamiento de entrada/salida
0404  F889 CDDCF8      CALL    Select$Routine  ;Seleccionar rutina CONOUT correcta
0405                                                ;Estas rutinas devuelven control directamente
0406                                                ;  a la que llamó a CONOUT
0407  F88C 38F9        DW      Teletype$Output         ;00 <- bits 1,0 del IOBYTE
0408  F88E 3EF9        DW      Terminal$Output         ;01
0409  F890 44F9        DW      Communication$Output    ;10
0410  F892 4AF9        DW      Dummy$Output            ;11
0411                ;
0412                ;
0413                ;
0414                LISTST:             ;Estado del dispositivo de listado (impresión, salida)
0415                                    ;Entrada directamente desde el vector JMP del BIOS;
0416                                    ;  devuelve en el registro A el estado del disp. de listado
0417                                    ;  que indica cuándo el dispositivo de listado puede aceptar
0418                                    ;  otro carácter de salida. Los bits 7, 6 del IOBYTE determinan
0419                                    ;  el dispositivo físico a utilizar
0420                                    ;
0421                                    ;A = 00H  (indicador de cero a uno) no puede aceptar datos
0422                                    ;A = 0FFH (indicador de cero a cero) puede aceptar datos
0423                                    ;
```

Figura 6-4. (Continuación.)

```
0424                                    ;La documentación de Digital Research indica
0425                                    ;  que se puede volver siempre con A = 00H (no pueden
0426                                    ;  aceptarse datos) si no se desea implementar
0427                                    ;  la rutina LISTST. Ello no es cierto.
0428                                    ;Si no se desea implementar la rutina,
0429                                    ;  LISTST siempre se volverá con A = 0FFH (puede aceptar datos).
0430                                    ;El controlador de impresora realiza
0431                                    ;  tareas que potencialmente cuelgan del sistema
0432                                    ;
0433  F894 CD9CF8       CALL    Get$List$Status ;Devuelve A = cero o distinto de cero
0434                                    ;  de acuerdo con el estado, se debe
0435                                    ;  convertir para parámetros convencionales
0436  F897 B7           ORA     A               ;Fijar marcas para reflejar el estado
0437  F898 C8           RZ                      ;Si 0, no pueden aceptarse datos para salida
0438  F899 3EFF         MVI     A,0FFH          ;En caso contrario devuelve A = 0FFH para indicar
0439  F89B C9           RET                     ;  que puede aceptar datos para salida
0440                ;
0441                Get$List$Status:
0442  F89C 3A0300       LDA     IOBYTE          ;Obtención byte de redireccionamiento de entrada/salida
0443  F89F 07           RLC                     ;Transferencia de los bits 7,6 a los 1,0
0444  F8A0 07           RLC
0445  F8A1 CDDCF8       CALL    Select$Routine  ;Selección de la rutina apropiada
0446                                                      ;Estas rutinas devuelven control directamente
0447                                                      ; a la que llamó a Get$List$Status
0448  F8A4 0BF9         DW      Teletype$Out$Status             ;00 <- bits 1,0 del IOBYTE
0449  F8A6 11F9         DW      Terminal$Out$Status             ;01
0450  F8A8 17F9         DW      Communication$Out$Status        ;10
0451  F8AA 1DF9         DW      Dummy$Out$Status                ;11
0452
0453                ;
0454                ;
0455                ;
0456                LIST:               ;Salida de impresión
0457                                    ;Entrada directamente desde vector JMP del BIOS;
0458                                    ; lanza a salida el carácter dato en el registro C
0459                                    ; sobre el dispositivo apropiado de acuerdo
0460                                    ; con los bits 7,6 del IOBYTE
0461
0462  F8AC 3A0300       LDA     IOBYTE          ;Obtención byte de redireccionamiento de lectura/escritura
0463  F8AF 07           RLC                     ;Transferencia de los bits 7,6 a 1,0
0464  F8B0 07           RLC
0465  F8B1 CDDCF8       CALL    Select$Routine          ;Selección de la rutina correcta de listado
0466                                                    ;Estas rutinas devuelven el control directamente
0467                                                    ; a la que llamó a LIST
0468  F8B4 38F9         DW      Teletype$Output         ;00 <- bits 1,0 del IOBYTE
0469  F8B6 3EF9         DW      Terminal$Output         ;01
0470  F8B8 44F9         DW      Communication$Output    ;10
0471  F8BA 4AF9         DW      Dummy$Output            ;11
0472
0473                ;
0474                ;
0475                ;
0476                PUNCH:              ;Salida por perforadora
0477                                    ;Entrada directamente desde vector JMP del BIOS;
0478                                    ; lanza a salida carácter dato en el registro C
0479                                    ; sobre el dispositivo apropiado de acuerdo
0480                                    ; con los bits 5,4 del IOBYTE
0481
0482  F8BC 3A0300       LDA     IOBYTE          ;Obtención byte de redireccionamiento de lectura/escritura
0483  F8BF 0F           RRC                     ;Transferencia de los bits 5,4 a 2,1
0484  F8C0 0F           RRC
0485  F8C1 0F           RRC
0486  F8C2 CDDDF8       CALL    Select$Routine$21       ;Selección de la rutina de perforación correcta
0487                                                    ;Estas rutinas devuelven el control directamente
0488                                                    ; a la que llamó a PUNCH
0489  F8C5 38F9         DW      Teletype$Output         ;00 <- bits 1,0 del IOBYTE
0490  F8C7 4AF9         DW      Dummy$Output            ;01
0491  F8C9 44F9         DW      Communication$Output    ;10
0492  F8CB 3EF9         DW      Terminal$Output         ;11
0493                ;
0494                ;
0495                ;
0496                READER:             ;Entrada por lectora
0497                                    ;Entrada directamente desde vector JMP del BIOS;
0498                                    ; entra en el registro A el carácter dato siguiente
0499                                    ; desde el dispositivo de lectura
```

Figura 6-4. (Continuación.)

```
0500                                    ;El dispositivo apropiado se selecciona de acuerdo con
0501                                    ; los bits 3,2 del IOBYTE
0502                                    ;
0503  F8CD 3A0300      LDA     IOBYTE            ;Obtención byte de redireccionamiento de lec./esc.
0504  F8D0 0F          RRC                       ;Transferencia de bits 3,2 a 2,1
0505  F8D1 CDDDF8      CALL    Select$Routine$21  ;Seleccionar rutina READER correcta
0506                                             ;Estas rutinas devolverán el control directamente
0507                                             ; a la que llamó a READER
0508  F8D4 38F9        DW      Teletype$Output    ;00 <- bits 1,0 del IOBYTE
0509  F8D6 4AF9        DW      Dummy$Output       ;01
0510  F8D8 44F9        DW      Communication$Output ;10
0511  F8DA 3EF9        DW      Terminal$Output    ;11
0512
0513                ;
0514                ;
0515                ;
0516                Select$Routine:           ;Transfiere el control a una dirección especifica
0517                                          ; siguiendo su dirección de llamada de acuerdo
0518                                          ; con el valor de los bits 1,0 de A
0519  F8DC 07           RLC                   ;Desplazamiento de los valores seleccionados a los bits 2,1
0520                                          ; para realizar aritmética de palabras
0521                ;
0522                Select$Routine$21:        ;Punto de entrada para seleccionar la rutina de seleccion
0523                                          ; los bits están ya en 2,1
0524  F8DD E606        ANI     0000$0110B    ;Aislar bits 2,1
0525  F8DF E3          XTHL                  ;HL -> primera palabra de dirección después
0526                                          ; de instrucción de llamada
0527  F8E0 5F          MOV     E,A           ;Añadir el valor de selección a tabla base
0528  F8E1 1600        MVI     D,0           ; de direcciones
0529  F8E3 19          DAD     D             ;HL -> dirección de rutina seleccionada
0530                                          ;Obtención de dirección de rutina en HL
0531  F8E4 7E          MOV     A,M           ;Byte LS
0532  F8E5 23          INX     H             ;HL -> byte MS
0533  F8E6 66          MOV     H,M           ;Byte MS
0534  F8E7 6F          MOV     L,A           ;HL -> rutina
0535  F8E8 E3          XTHL                  ;Cabecera de pila -> rutina
0536  F8E9 C9          RET                   ;Transferencia a rutina seleccionada
0537                ;
0538                ;
0539                ;
0540                ;  Input/Output Equates
0541                ;
0542  00ED =        Teletype$Status$Port              EQU     0EDH
0543  00EC =        Teletype$Data$Port          EQU   0ECH
0544  0001 =        Teletype$Output$Ready             EQU     0000$0001B      ;Máscara de estado
0545  0002 =        Teletype$Input$Ready              EQU     0000$0010B      ;Máscara de estado
0546                ;
0547  0001 =        Terminal$Status$Port              EQU     01H
0548  0002 =        Terminal$Data$Port          EQU   02H
0549  0001 =        Terminal$Output$Ready             EQU     0000$0001B      ;Máscara de estado
0550  0002 =        Terminal$Input$Ready              EQU     0000$0010B      ;Máscara de estado
0551                ;
0552  00ED =        Communication$Status$Port  EQU    0EDH
0553  00EC =        Communication$Data$Port           EQU     0ECH
0554  0001 =        Communication$Output$Ready EQU    0000$0001B    ;Máscara de estado
0555  0002 =        Communication$Input$Ready  EQU    0000$0010B    ;Máscara de estado
0556                ;
0557  00DF =        Communication$Baud$Mode           EQU     0DFH            ;Selección de modo
0558  00DE =        Communication$Baud$Rate           EQU     0DEH            ;Selección de velocidad
0559                ;
0560                ;
0561                ;  Tablas de control de dispositivo serie
0562                ;
0563                ;  Para reducir la cantidad de código ejecutable, se utiliza
0564                ;  el mismo programa de control de bajo nivel para todos los puertos serie.
0565                ;  Al entrar al controlador de bajo nivel, HL apunta a la tabla
0566                ;  de control apropiada
0567                ;
0568                Teletype$Table:
0569  F8EA ED           DB      Teletype$Status$Port
0570  F8EB EC           DB      Teletype$Data$Port
0571  F8EC 01           DB      Teletype$Output$Ready
0572  F8ED 02           DB      Teletype$Input$Ready
0573                ;
0574                Terminal$Table:
0575  F8EE 01           DB      Terminal$Status$Port
```

Figura 6-4. (Continuación.)

```
0576  F8EF 02           DB      Terminal$Data$Port
0577  F8F0 01           DB      Terminal$Output$Ready
0578  F8F1 02           DB      Terminal$Input$Ready
0579                    ;
0580                    Communication$Table:
0581  F8F2 ED           DB      Communication$Status$Port
0582  F8F3 EC           DB      Communication$Data$Port
0583  F8F4 01           DB      Communication$Output$Ready
0584  F8F5 02           DB      Communication$Input$Ready
0585                    ;
0586                    ;
0587                    ;
0588                    ;
0589                    ;  Las rutinas siguientes son llamadas por Select$Routine
0590                    ;  para realizar la entrada/salida de bajo nivel:
0591                    ;
0592                    Teletype$In$Status:
0593  F8F6 21EAF8       LXI     H,Teletype$Table        ;HL -> tabla de control
0594  F8F9 C34BF9       JMP     Input$Status            ;Nótese el uso de JMP. Input$Status
0595                                                    ; ejecutará el RETorno
0596                    ;
0597                    Terminal$In$Status:
0598  F8FC 21EEF8       LXI     H,Terminal$Table        ;HL -> tabla de control
0599  F8FF C34BF9       JMP     Input$Status            ;Nótese el uso de JMP. Input$Status
0600                                                    ; ejecutará el RETorno
0601                    ;
0602                    Communication$In$Status:
0603  F902 21F2F8       LXI     H,Communication$Table   ;HL -> tabla de control
0604  F905 C34BF9       JMP     Input$Status            ;Nótese el uso de JMP. Input$Status
0605                                                    ; ejecutará el RETorno
0606                    ;
0607                    Dummy$In$Status:                ;Estado ficticio, siempre vuelve
0608  F908 3EFF         MVI     A,0FFH                  ; indicando dato preparado
0609  F90A C9           RET
0610                    ;
0611                    ;
0612                    Teletype$Out$Status:
0613  F90B 21EAF8       LXI     H,Teletype$Table        ;HL -> tabla de control
0614  F90E C356F9       JMP     Output$Status           ;Nótese el uso de JMP. Output$Status
0615                                                    ; ejecutará el RETorno
0616                    ;
0617                    Terminal$Out$Status:
0618  F911 21EEF8       LXI     H,Terminal$Table        ;HL -> tabla de control
0619  F914 C356F9       JMP     Output$Status           ;Nótese el uso de JMP. Output$Status
0620                                                    ; ejecutará el RETorno
0621                    ;
0622                    Communication$Out$Status:
0623  F917 21F2F8       LXI     H,Communication$Table   ;HL -> tabla de control
0624  F91A C356F9       JMP     Output$Status           ;Nótese el uso de JMP. Output$Status
0625                                                    ; ejecutará el RETorno
0626                    ;
0627                    Dummy$Out$Status:               ;Estado ficticio, siempre vuelve
0628  F91D 3EFF         MVI     A,0FFH                  ; indicando preparado para salida
0629  F91F C9           RET
0630                    ;
0631                    ;
0632                    Teletype$Input:
0633  F920 21EAF8       LXI     H,Teletype$Table        ;HL -> tabla de control
0634  F923 C360F9       JMP     Input$Data              ;Nótese el uso de JMP. Input$Data
0635                                                    ; ejecutará el RETorno
0636                    ;
0637                    Terminal$Input:
0638  F926 21EEF8       LXI     H,Terminal$Table        ;HL -> tabla de control
0639                                                    ; ejecutará el RETorno
0640  F929 CD60F9       CALL    Input$Data              ;** Caso especial **
0641                                                    ;Input$Data volverá aquí ya que
0642  F92C E67F         ANI     7FH                     ; bit de paridad puesto a 0
0643  F92E C9           RET
0644                    ;
0645                    Communication$Input:
0646  F92F 21F2F8       LXI     H,Communication$Table   ;HL -> tabla de control
0647  F932 C360F9       JMP     Input$Data              ;Nótese el uso de JMP. Input$Data
0648                                                    ; ejecutará el RETorno
0649                    ;
0650                    Dummy$Input:                    ;Estado ficticio, siempre vuelve
0651  F935 3E1A         MVI     A,1AH                   ; indicando final de fichero CP/M
```

Figura 6-4. (Continuación.)

```
0652  F937 C9            RET
0653                    ;
0654                    ;
0655                    ;
0656                    ;
0657                    Teletype$Output:
0658  F938 21EAF8       LXI     H,Teletype$Table        ;HL -> tabla de control
0659  F93B C370F9       JMP     Output$Data             ;Nótese el uso del JMP. Output$Data
0660                                                    ; para ejecutar el RETorno
0661                    ;
0662                    Terminal$Output:
0663  F93E 21EEF8       LXI     H,Terminal$Table        ;HL -> tabla de control
0664                                                    ; que ejecutará el RETorno
0665  F941 C370F9       JMP     Output$Data             ;Nótese el uso del JMP. Output$Data
0666                                                    ; para ejecutar el RETorno
0667                    ;
0668                    Communication$Output:
0669  F944 21F2F8       LXI     H,Communication$Table   ;HL -> tabla de control
0670  F947 C370F9       JMP     Output$Data             ;Nótese el uso del JMP. Output$Data
0671                                                    ; para ejecutar el RETorno
0672                    ;
0673                    Dummy$Output:                   ;Salida ficticia, siempre descarta
0674  F94A C9           RET                             ; el carácter de salida
0675                    ;
0676                    ;
0677                    ;
0678                    ;
0679                    ;  Estos son controladores de bajo nivel de propósito general
0680                    ;  En entrada, HL apunta a la tabla de control apropiada
0681                    ;  En salida, el registro C contiene los datos-para-salida
0682                    ;
0683                    Input$Status:               ;Retorno con A = 00H si no hay datos de entrada,
0684                                                ; en caso contrario A distinto de cero.
0685  F94B 7E           MOV     A,M                 ;Obtención estado del puerto
0686  F94C 3250F9       STA     Input$Status$Port       ;*** Código automodificable ***
0687  F94F DB           DB      IN                  ;Entrada a A del puerto en estado correcto
0688                    ;
0689                    Input$Status$Port:
0690  F950 00           DB      00                  ;<- Fijar anterior
0691  F951 23           INX     H                   ;Cambiar HL para que apunte a la máscara de datos de
0692  F952 23           INX     H                   ; entrada
0693  F953 23           INX     H
0694  F954 A6           ANA     M                   ;Máscara con estado de entrada
0695  F955 C9           RET
0696                    ;
0697                    ;
0698                    Output$Status:              ;Retorno con A = 00H si no está preparado para salida,
0699                                                ; en caso contrario A distinto de cero
0700  F956 7E           MOV     A,M                 ;Obtención estado del puerto
0701  F957 325BF9       STA     Output$Status$Port      ;*** Código automodificable ***
0702  F95A DB           DB      IN                  ;Entrada a A del puerto en estado correcto
0703                    ;
0704                    Output$Status$Port:
0705  F95B 00           DB      00                  ;<- Fijar anterior
0706  F95C 23           INX     H                   ;Cambiar HL para que apunte a la máscara de datos de salida
0707  F95D 23           INX     H
0708  F95E A6           ANA     M                   ;Máscara con estado de salida
0709  F95F C9           RET
0710                    ;
0711                    ;
0712                    Input$Data:                 ;Retorno con el siguiente carácter-dato en A
0713                                                ;Tiempo de espera para que la rutina de estado
0714                                                ; indique datos-de-entrada
0715  F960 E5           PUSH    H                   ;Almacenar apuntador a tabla de control
0716  F961 CD4BF9       CALL    Input$Status        ;Obtención del estado de entrada en el indicador de cero
0717  F964 E1           POP     H                   ;Recuperar apuntador a tabla de control
0718  F965 CA60F9       JZ      Input$Data          ;Espera hasta que existan datos de entrada
0719  F968 23           INX     H                   ;HL -> puerto de datos
0720  F969 7E           MOV     A,M                 ;Obtención puerto de datos
0721  F96A 326EF9       STA     Input$Data$Port     ;*** Código automodificable ***
0722  F96D DB           DB      IN                  ;Entrada a A desde el puerto de datos correcto
0723                    ;
0724                    Input$Data$Port:
0725  F96E 00           DB      0                 ;<- Fijar anterior
0726  F96F C9           RET
0727                    ;
```

Figura 6-4. (Continuación.)

```
0728                    ;
0729                    Output$Data:                    ;Salida de los caracteres de datos en el registro C
0730                                                    ;Espera para que la rutina de estado indique dispositivo
0731                                                    ; preparado para aceptar otro carácter
0732  F970 E5               PUSH    H                  ;Almacenamiento apuntador a tabla de control
0733  F971 CD56F9           CALL    Output$Status      ;Obtención estado de salida en marca-cero
0734  F974 E1               POP     H                  ;Recuperar apuntador a tabla de control
0735  F975 CA70F9           JZ      Output$Data        ;Espera hasta preparado-para-salida
0736  F978 23               INX     H                  ;HL -> puerto de salida
0737  F979 7E               MOV     A,M                ;Obtención puerto de salida
0738  F97A 327FF9           STA     Output$Data$Port       ;*** Código automodificable ***
0739  F97D 79               MOV     A,C                ;Obtención carácter de datos para lanzar a salida
0740  F97E D3               DB      OUT                ;Salida de datos al puerto correcto
0741                    ;
0742                    Output$Data$Port:
0743  F97F 00               DB      0                  ;<- Fijar lo anterior
0744  F980 C9               RET
0745                    ;
0746                    ;
0747                    ;  Controladores de disquete de alto nivel
0748                    ;
0749                    ;  Estos disquetes realizan las siguientes funciones:
0750                    ;
0751                    ;  SELDSK  Selecciona el disco especificado y devuelve la dirección
0752                    ;          de la tabla cabecera de parámetros de disco
0753                    ;  SETTRK  Fija el número de pista para la próxima lectura o escritura
0754                    ;  SETSEC  Fija el número de sector para la próxima lectura o escritura
0755                    ;  SETDMA  Fija la dirección DMA (lectura/escritura) para la próxima operación
0756                    ;  SECTRAN Traduce un número de sector lógico a físico
0757                    ;  HOME    Fija la pista a cero de forma que la lectura o escritura siguiente
0758                    ;          se realizará en la pista cero
0759                    ;
0760                    ;  En resumen, los controladores de alto nivel son los responsables
0761                    ;  de hacer que los disquetes de 5 1/4" que utilizan sectores de 512 bytes
0762                    ;  aparezcan a CP/M como si utilizaran sectores de 128 bytes. Realizan esta
0763                    ;  operación utilizando el programa llamado de bloqueo/desbloqueo descrito
0764                    ;  con mayor detalle en las páginas siguientes de este listado,
0765                    ;  justo antes que el propio programa.
0766                    ;
0767                    ;
0768                    ;
0769                    ;  Tablas de parámetros de disco
0770                    ;
0771                    ;  Como se mencionó en el capítulo 3, ésta describe las
0772                    ;  características físicas de los controladores de disco. En este
0773                    ;  ejemplo de BIOS hay dos tipos de controladores, los estándar
0774                    ;  de 8" simple cara, simple densidad y los de 5 1/4" de doble cara,
0775                    ;  doble densidad.
0776                    ;
0777                    ;  Los disquetes estándar de 8" no precisan del uso del programa
0778                    ;  de bloqueo/desbloqueo, pero sí los de 5 1/4". Un byte
0779                    ;  adicional ha sido añadido al bloque de parámetros del disco para indicar
0780                    ;  al controlador el tipo físico del disquete lógico y si necesita
0781                    ;  o no desbloqueo.
0782                    ;
0783                    ;
0784                    ;  Tablas de definición del disco
0785                    ;
0786                    ;  Consisten en cabeceras de parámetros de disco con una entrada
0787                    ;  por cada unidad lógica de disco y bloque de parámetros,
0788                    ;  con el mismo bloque de parámetros para varios discos lógicos
0789                    ;  o uno por disco lógico.
0790                    ;
0791                    ;
0792                    Disk$Parameter$Headers:                 ;Descrito en el capítulo 3
0793                    ;
0794                                    ;Logical Disk A: (5 1/4" Diskette)
0795  F981 6BFB             DW      Floppy$5$Skewtable        ;Tabla de desplazamiento de 5 1/4"
0796  F983 0000000000       DW      0,0,0                     ;Reservado para CP/M
0797  F989 C1F9             DW      Directory$Buffer
0798  F98B 42FA             DW      Floppy$5$Parameter$Block
0799  F98D 61FA             DW      Disk$A$Workarea
0800  F98F C1FA             DW      Disk$A$Allocation$Vector
0801                    ;
0802                                    ;Logical Disk B: (5 1/4" Diskette)
0803  F991 6BFB             DW      Floppy$5$Skewtable        ;Comparte misma tabla de desplaz. que A:
```

Figura 6-4. (Continuación.)

```
0804  F993 0000000000    DW      0,0,0                   ;Reservado para CP/M
0805  F999 C1F9          DW      Directory$Buffer        ;Comparte misma memoria con A:
0806  F99B 42FA          DW      Floppy$5$Parameter$Block ;Igual DPB que A:
0807  F99D 81FA          DW      Disk$B$Workarea         ;Area de trabajo privada
0808  F99F D7FA          DW      Disk$B$Allocation$Vector ;Vector de ubicación privado
0809                ;
0810                                ;Logical Disk C: (8" Floppy)
0811  F9A1 B3FB          DW      Floppy$8$Skewtable      ;Tabla de desplazamiento de 8"
0812  F9A3 0000000000    DW      0,0,0                   ;Reservado para CP/M
0813  F9A9 C1F9          DW      Directory$Buffer        ;Comparte misma memoria intermedia que A:
0814  F9AB 52FA          DW      Floppy$8$Parameter$Block
0815  F9AD A1FA          DW      Disk$C$Workarea         ;Area de trabajo privada
0816  F9AF EDFA          DW      Disk$C$Allocation$Vector ;Vector de ubicación privado
0817                ;
0818                                ;Logical Disk D: (8" Floppy)
0819  F9B1 6BFB          DW      Floppy$5$Skewtable      ;Comparte misma tabla de desplaz. que A:
0820  F9B3 0000000000    DW      0,0,0                   ;Reservado para CP/M
0821  F9B9 C1F9          DW      Directory$Buffer        ;Comparte misma memoria intermedia que A:
0822  F9BB 52FA          DW      Floppy$8$Parameter$Block ;Igual DPB que C:
0823  F9BD B1FA          DW      Disk$D$Workarea         ;Area de trabajo privada
0824  F9BF 0CFB          DW      Disk$D$Allocation$Vector ;Vector de ubicación privado
0825
0826                ;
0827                ;
0828  F9C1          Directory$Buffer:  DS      128
0829                ;
0830                ;
0832                ;
0833                ;  Tipos de disco
0834                ;
0835  0001 =        Floppy$5   EQU     1       ;Minidisquete de 5 1/4''
0836  0002 =        Floppy$8   EQU     2       ;Disquete de 8" (SS SD)
0837                ;
0838                ;  Blocking/deblocking indicator
0839                ;
0840  0080 =        Need$Deblocking    EQU     1000$0000B      ;Tamaño de sector > 128 bytes
0841                ;
0842                ;
0843                ;  Disk parameter blocks
0844                ;
0845                ;  5 1/4" mini floppy
0846                ;
0847                                        ;Byte extra con prefijo para indicar tipo de disco
0848                                        ; y necesidad de bloqueo
0849  FA41 81           DB      Floppy$5 + Need$Deblocking
0850                Floppy$5$Parameter$Block:
0851  FA42 4800         DW      72                      ;Sectores de 128 bytes por pista
0852  FA44 04           DB      4                       ;Desplazamiento de bloque
0853  FA45 0F           DB      15                      ;Máscara de bloques
0854  FA46 01           DB      1                       ;Máscara de extensión
0855  FA47 AE00         DW      174                     ;Número máximo de bloque
0856  FA49 7F00         DW      127                     ;Número de entradas al directorio - 1
0857  FA4B C0           DB      1100$0000B              ;Mapa de bits para reservar un bloque
0858  FA4C 00           DB      0000$0000B              ; por cada directorio de ficheros
0859  FA4D 2000         DW      32                      ;Cambio de tamaño de área de trabajo en disco
0860  FA4F 0100         DW      1                       ;Número de pistas antes del directorio
0861                ;
0862                ;
0863                ;  Disquete de 8'' normalizado
0864                                        ;Byte extra con prefijo al DPB
0865                                        ; para esta versión de BIOS
0866  FA51 02           DB      Floppy$8                ;Indica tipo de disco y el hecho
0867                                        ; de no necesitar desbloqueo
0868                Floppy$8$Parameter$Block:
0869  FA52 1A00         DW      26                      ;Sectores por pista
0870  FA54 03           DB      3                       ;Desplazamiento de bloque
0871  FA55 07           DB      7                       ;Máscara de bloques
0872  FA56 00           DB      0                       ;Máscara de extensión
0873  FA57 F200         DW      242                     ;Número máximo de bloque
0874  FA59 3F00         DW      63                      ;Número de entradas al directorio - 1
0875  FA5B C0           DB      1100$0000B              ;Mapa de bits para reservar dos bloques
0876  FA5C 00           DB      0000$0000B              ; por cada directorio de ficheros
0877  FA5D 1000         DW      16                      ;Cambio de tamaño de área de trabajo en disco
0878  FA5F 0200         DW      2                       ;Número de pistas antes del directorio
0879                ;
0880                ;
```

Figura 6-4. (Continuación.)

```
0881                    ;  Areas de trabajo del disco
0882                    ;
0883                    ;  Son utilizadas por el BDOS para detectar cambios de disquete
0884                    ;  no esperados. El BDOS pondrá automáticamente en estado
0885                    ;  de sólo lectura al disquete que ha sido cambiado.
0886                    ;
0887  FA61              Disk$A$Workarea:    DS      32      ; A:
0888  FA81              Disk$B$Workarea:    DS      32      ; B:
0889  FAA1              Disk$C$Workarea:    DS      16      ; C:
0890  FAB1              Disk$D$Workarea:    DS      16      ; D:
0891                    ;
0892                    ;
0893                    ;  Vectores de ubicación del disco
0894                    ;
0895                    ;  Son utilizados por el BDOS para mantener un mapa de bits
0896                    ;  de cuáles son los bloques usados y cuáles no.
0897                    ;  Se utiliza un byte por cada 8 bloques, de ahí su expresión
0898                    ;  en la forma (bloques/8)+1.
0899                    ;
0900  FAC1              Disk$A$Allocation$Vector    DS      (174/8)+1       ; A:
0901  FAD7              Disk$B$Allocation$Vector    DS      (174/8)+1       ; B:
0902                    ;
0903  FAED              Disk$C$Allocation$Vector    DS      (242/8)+1       ; C:
0904  FB0C              Disk$D$Allocation$Vector    DS      (242/8)+1       ; D:
0905                    ;
0906                    ;
0907  0004 =            Number$of$Logical$Disks             EQU     4
0908                    ;
0909                    ;
0910                    SELDSK:                         ;Disco seleccionado en C
0911                                                    ;C = 0 para unidad A, 1 para B, etc.
0912                                                    ;Retorno de la dirección de la cabecera
0913                                                    ; de parámetros del disco en HL,
0914                                                    ; o 0000H si no existe el disco seleccionado.
0915
0916  FB2B 210000           LXI     H,0             ;Suposición de error
0917  FB2E 79               MOV     A,C             ;Comprobación de que el disco que se demanda es válido
0918  FB2F FE04             CPI     Number$of$Logical$Disks
0919  FB31 D0               RNC                     ;Retorno si > que el máximo número de discos
0920                    ;
0921  FB32 32EAFB           STA     Selected$Disk   ;Almacenar número de disco seleccionado
0922                                                ;Activación para retorno de dirección DPH
0923  FB35 6F               MOV     L,A             ;Componer valor disco en palabra
0924  FB36 2600             MVI     H,0
0925                                                ;Calcular el desplazamiento inferior cabecera de tabla
0926                                                ; de parámetros de disco multiplicando por la
0927                                                ; longitud de la cabecera (16 bytes)
0928  FB38 29               DAD     H               ; *2
0929  FB39 29               DAD     H               ; *4
0930  FB3A 29               DAD     H               ; *8
0931  FB3B 29               DAD     H               ; *16
0932  FB3C 1181F9           LXI     D,Disk$Parameter$Headers        ;Obtención dirección base
0933  FB3F 19               DAD     D               ;DE -> DPH apropiado
0934  FB40 E5               PUSH    H               ;Almacenar dirección DPH
0935                    ;
0936                                                ;Acceso al bloque de parámetros de disco
0937                                                ; para extracción del byte especial de prefijo
0938                                                ; que identifica el tipo y si es
0939                                                ; necesario desbloqueo
0940
0941  FB41 110A00           LXI     D,10            ;Obtención del desplazamiento apuntador DPB en DPH
0942  FB44 19               DAD     D               ;DE -> dirección DPB en DPH
0943  FB45 5E               MOV     E,M             ;Obtención dirección DPB en DE
0944  FB46 23               INX     H
0945  FB47 56               MOV     D,M
0946  FB48 EB               XCHG                    ;DE -> DPB
0947  FB49 2B               DCX     H               ;DE -> byte prefijo
0948  FB4A 7E               MOV     A,M             ;Obtención byte prefijo
0949  FB4B E60F             ANI     0FH             ;Aislar tipo de disco
0950  FB4D 32FAFB           STA     Disk$Type       ;Almacenamiento para utilizar en controlador de bajo nivel
0951  FB50 7E               MOV     A,M             ;Obtención de una nueva copia del byte de prefijo
0952  FB51 E680             ANI     Need$Deblocking         ;Aislamiento de la marca de desbloqueo
0953  FB53 32F9FB           STA     Deblocking$Required     ;Almacenar para utilizar en controlador de bajo nivel
0954  FB56 E1               POP     H               ;Recuperación apuntador a DPH
0955  FB57 C9               RET
0956                    ;
```

Figura 6-4. (Continuación.)

```
0957                    ;
0958                    ;  Fijación de pista lógica para la siguiente lectura o escritura
0959                    ;
0960                    SETTRK:
0961  FB58 60              MOV     H,B             ;Pista seleccionada en BC en entrada
0962  FB59 69              MOV     L,C
0963  FB5A 22EBFB          SHLD    Selected$Track  ;Almacenar para controlador de bajo nivel
0964  FB5D C9              RET
0965                    ;
0966                    ;
0967                    ;  Fijación de sector lógico para la siguiente lectura o escritura
0968                    ;
0969                    ;
0970                    SETSEC:                     ;Sector lógico en C en entrada
0971  FB5E 79              MOV     A,C
0972  FB5F 32EDFB          STA     Selected$Sector ;Almacenamiento para controlador de bajo nivel
0973  FB62 C9              RET
0974                    ;
0975                    ;
0976                    ;  Fijación de dirección de DMA (entrada/salida) para siguiente lectura/escritura
0977                    ;
0978  FB63 0000         DMA$Address:        DW      0       ;Dirección DMA
0979                    ;
0980                    SETDMA:                     ;Dirección en BC en entrada
0981  FB65 69              MOV     L,C             ;Puesta en HL para almacenar
0982  FB66 60              MOV     H,B
0983  FB67 2263FB          SHLD    DMA$Address     ;Almacenar para controlador de bajo nivel
0984  FB6A C9              RET
0985                    ;
0986                    ;
0987                    ;  Traducción de número de sector lógico a físico
0988                    ;
0989                    ;  Tablas de traducción de sectores
0990                    ;  Estas tablas están indexadas utilizando el número de sector lógico, y
0991                    ;  contienen el correspondiente número de sector físico
0992                    ;
0993                    Floppy$5$Skewtable:         ;Cada sector físico contiene
0994                                                ; 4 sectores de 128 bytes.
0995                        ;       Físicos 128b    Lógicos 128b        Físicos 512b
0996  FB6B 00010203         DB      00,01,02,03     ;00,01,02,03            0   )
0997  FB6F 10111213         DB      16,17,18,19     ;04,05,06,07            4   )
0998  FB73 20212223         DB      32,33,34,35     ;08,09,10,11            8   )
0999  FB77 0C0D0E0F         DB      12,13,14,15     ;12,13,14,15            3   ) Cabeza
1000  FB7B 1C1D1E1F         DB      28,29,30,31     ;16,17,18,19            7   )   0
1001  FB7F 08090A0B         DB      08,09,10,11     ;20,21,22,23            2   )
1002  FB83 18191A1B         DB      24,25,26,27     ;24,25,26,27            6   )
1003  FB87 04050607         DB      04,05,06,07     ;28,29,30,31            1   )
1004  FB8B 14151617         DB      20,21,22,23     ;32,33,34,35            5   )
1005                    ;
1006  FB8F 24252627         DB      36,37,38,39     ;36,37,38,39            0   ]
1007  FB93 34353637         DB      52,53,54,55     ;40,41,42,43            4   ]
1008  FB97 44454647         DB      68,69,70,71     ;44,45,46,47            8   ]
1009  FB9B 30313233         DB      48,49,50,51     ;48,49,50,51            3   ] Cabeza
1010  FB9F 40414243         DB      64,65,66,67     ;52,53,54,55            7   ]   1
1011  FBA3 2C2D2E2F         DB      44,45,46,47     ;56,57,58,59            2   ]
1012  FBA7 3C3D3E3F         DB      60,61,62,63     ;60,61,62,63            6   ]
1013  FBAB 28292A2B         DB      40,41,42,43     ;64,65,66,67            1   ]
1014  FBAF 38393A3B         DB      56,57,58,59     ;68,69,70,71            5   ]
1015                    ;
1016                    ;
1017                    Floppy$8$Skewtable:         ;Controlador de 8" normalizado
1018                        ;       01,02,03,04,05,06,07,08,09,10     Sectores lógicos
1019  FBB3 01070D1319       DB      01,07,13,19,25,05,11,17,23,03    ;Sectores físicos
1020                        ;
1021                        ;       11,12,13,14,15,16,17,18,19,20     Sectores lógicos
1022  FBBD 090F150208       DB      09,15,21,02,08,14,20,26,06,12    ;Sectores físicos
1023                        ;
1024                        ;       21,22,23,24,25,26         Sectores lógicos
1025  FBC7 1218040A10       DB      18,24,04,10,16,22        ;Sectores físicos
1026
1027                    ;
1028                    SECTRAN:                    ;Traducción de sector lógico a físico
1029                                                ;En entrada, BC = número de sector lógico
1030                                                ;            DE -> tabla de desplazamiento apropiada
1031                                                ;
1032                                                ;en salida, HL = número de sector físico
```

Figura 6-4. (Continuación.)

```
1033  FBCD EB           XCHG                    ;HL -> base de tabla de desplazamiento
1034  FBCE 09           DAD     B               ;Añadir en número de sector lógico
1035  FBCF 6E           MOV     L,M             ;Obtención número de sector físico
1036  FBD0 2600         MVI     H,0             ;Componer en un valor de 16 bytes
1037  FBD2 C9           RET
1038                  ;
1039                  ;
1040                  ;
1041                  HOME:                     ;Devolver el disco lógico seleccionado a la pista 0
1042                                            ; Antes de realizar la operación debe comprobarse
1043                                            ; que la memoria intermedia de disco físico no tiene
1044                                            ; información que se debe escribir. Está indicado por la
1045                                            ; marca Must$Write$Buffer fijada en el código de
1046                                            ; desbloqueo
1047                                            ;
1048  FBD3 3AE9FB       LDA     Must$Write$Buffer       ;Comprobación de si la memoria intermedia física
1049  FBD6 B7           ORA     A                       ;  debe ser escrita fuera del disco
1050  FBD7 C2DDFB       JNZ     HOME$No$Write
1051  FBDA 32E8FB       STA     Data$In$Disk$Buffer     ;No, indicación de disco
1052                                                    ;  desocupado
1053                  HOME$No$Write:
1054  FBDD 0E00         MVI     C,0                     ;Envío a pista 0 (lógicamente --
1055  FBDF CD58FB       CALL    SETTRK                  ;  la operación no ocurre realmente en el disco)
1056  FBE2 C9           RET
1057
1058                  ;
1059                  ;  Los datos escritos o leídos en la unidad de disquetes son transferidos
1060                  ;  vía una memoria intermedia que es en realidad de 512 bytes (ha sido
1061                  ;  declarada al principio del BIOS y mantiene el programa de inicialización de
1062                  ;  "una-sola-vez" que se utiliza en el proceso de arranque en frío).
1063                  ;
1064                  ;  El programa de bloqueo/desbloqueo intenta minimizar la cantidad
1065                  ;  de entrada/salida del disco almacenando en la memoria intermedia física
1066                  ;  el disco, pista y sector físico residente en curso.
1067                  ;  Si se da una petición de lectura de un sector CP/M de 128 bytes que está
1068                  ;  aún en la memoria intermedia física, no es necesario, en este caso, un acceso al disco.
1069                  ;
1070                  ;
1071  0800 =          Allocation$Block$Size    EQU     2048
1072  0012 =          Physical$Sec$Per$Track   EQU     18
1073  0004 =          CPM$Sec$Per$Physical     EQU     Physical$Sector$Size/128
1074  0048 =          CPM$Sec$Per$Track        EQU     CPM$Sec$Per$Physical*Physical$Sec$Per$Track
1075  0003 =          Sector$Mask              EQU     CPM$Sec$Per$Physical-1
1076  0002 =          Sector$Bit$Shift         EQU     2       ;LOG2(CPM$Sec$Per$Physical)
1077                  ;
1078                                  ;Son valores manejados por el BDOS cuando hace una
1079                                  ;  operación WRITE
1080                                  ;La asignación/no asignación indica cuándo el BDOS está
1081                                  ;  activado para escribir en un bloque no asignado
1082                                  ;  (sólo lo indica para la escritura del primer sector
1083                                  ;  de 128 bytes) o en un bloque que ha sido ya asignado
1084                                  ;  a un fichero
1085                                  ;El BDOS indica también si está activado para escribir
1086                                  ;  el directorio de ficheros
1087                                  ;
1088  0000 =          Write$Allocated          EQU     0
1089  0001 =          Write$Directory          EQU     1
1090  0002 =          Write$Unallocated        EQU     2
1091                  ;
1092  FBE3 00         Write$Type:              DB      0       ;Contiene el tipo de escritura
1093                                                          ;  indicado por el BDOS
1094                  ;
1095                  ;
1096                  In$Buffer$Dk$Trk$Sec:                   ;Variables para el sector físico en curso
1097                                                          ; en Disk$Buffer en la memoria
1098  FBE4 00         In$Buffer$Disk:          DB      0      ; Son cambiadas y comparadas
1099  FBE5 0000       In$Buffer$Track:         DW      0      ; como grupo de forma que
1100  FBE7 00         In$Buffer$Sector:        DB      0      ; estas líneas no se alteran
1101                  ;
1102  FBE8 00         Data$In$Disk$Buffer:     DB      0      ;Cuando es distinto de cero la mem. inter.
1103                                                          ;  del disco contiene datos del disco.
1104  FBE9 00         Must$Write$Buffer:       DB      0      ;Distinto de cero cuando los datos han sido
1105                                                          ;  escritos en Disk$Buffer pero no han
1106                                                          ;  sido escritos todavía fuera del disco
1107                  ;
1108                  Selected$Dk$Trk$Sec:    ;Variables para el disco seleccionado, pista y sector
```

Figura 6-4. (Continuación.)

```
1109                                                    ; (Seleccionadas por SELDSK, SETTRK y SETSEC)
1110  FBEA 00       Selected$Disk:                  DB      0       ; Son comparadas y cambiadas
1111  FBEB 0000     Selected$Track:                 DW      0       ; como grupo de forma que
1112  FBED 00       Selected$Sector:                DB      0       ; no se altera el orden
1113
1114  FBEE 00       Selected$Physical$Sector:       DB      0       ;Sector físico seleccionado deducido del
1115                                                                ;  sector lógico seleccionado (CP/M),
1116                                                                ;  desplazándolo a la derecha
1117                                                                ;  el número de bits especificado
1118                                                                ;  por Sector$Bit$Shift
1119                  ;
1120  FBEF 00       Selected$Disk$Type:             DB      0       ;Fijación de SELDSK para indicar si se
1121                                                                ;  trata de un disquete de 8'' o de 5 1/4''
1122  FBF0 00       Selected$Disk$Deblock:          DB      0       ;Fijación de SELDSK para indicar cuándo es
1123                                                                ;  necesario desbloqueo
1124
1125
1126                Unallocated$Dk$Trk$Sec:                 ;Parámetros para escritura en un bloque
1127                                                        ;       desasignado previamente
1128  FBF1 00       Unallocated$Disk:               DB      0       ; Son cambiados y comparados
1129  FBF2 0000     Unallocated$Track:              DW      0       ; como grupo de forma que
1130  FBF4 00       Unallocated$Sector:             DB      0       ; no se alteran estas líneas
1131
1132  FBF5 00       Unallocated$Record$Count:       DB      0       ;Número de "registros" no asignados
1133                                                                ;  en el bloque en curso desasignado
1134                                                                ;  previamente
1135
1136  FBF6 00       Disk$Error$Flag:                DB      0       ;Distinto de cero para indicar un error
1137                                                                ;  que no puede ser recuperado por los
1138                                                                ;  controladores de disco. El BDOS lanzará
1139                                                                ;  un mensaje de "sector dañado"
1140                ;
1141                ;Flags used inside the deblocking code
1142
1143  FBF7 00       Must$Preread$Sector:             DB      0       ;Distinto de cero si un sector físico puede
1144                                                                ;  leerse en la memoria intermedia
1145                                                                ;  de disco bien antes de que tenga
1146                                                                ;  lugar la escritura de un bloque asignado
1147                                                                ;  o bien para una lectura de sector normal
1148                                                                ;  CP/M de 128 bytes
1149  FBF8 00       Read$Operation:                 DB      0       ;Distinto de cero cuando se lee
1150                                                                ;  un sector CP/M de 128 bytes
1151  FBF9 00       Deblocking$Required:            DB      0       ;Distinto de cero cuando el disco selec.
1152                                                                ;  precisa desbloqueo (fijado en SELDSK)
1153  FBFA 00       Disk$Type:                      DB      0       ;Indica si el disco seleccionado es de 8"
1154                                                                ;  o de 5 1/4" (fijado en SELDSK)
1155                ;
1156                ;
1157                ;  Lectura de un sector CP/M de 128 bytes especificado por las llamadas previas para
1158                ;  seleccionar disco y fijar pista y sector. El sector será leído en la
1159                ;  dirección especificada en la llamada previa para fijar la dirección de DMA.
1160                ;
1161                ;  Si se lee de una unidad de disco que usa sectores mayores que 128 bytes,
1162                ;  el programa de desbloqueo será utilizado para "desempaquetar" un sector
1163                ;  de 128 bytes del sector físico.
1164                READ:
1165  FBFB 3AF9FB       LDA     Deblocking$Required     ;Comprobación de si es preciso desbloqueo
1166  FBFE B7           ORA     A                       ;(la marca corres. fue fijada en llamada a SELDSK)
1167  FBFF CA52FD       JZ      Read$No$Deblock         ;No, utilización normal no desbloqueada
1168
1169                                ;El algoritmo de desbloqueo que se utiliza es tal
1170                                ;  que una operación de lectura puede
1171                                ;  ser vista hasta la transferencia de datos
1172                                ;  como si fuese la primera escritura
1173                                ;  a un bloque no asignado
1174  FC02 AF           XRA     A                       ;Puesta a cero del contador de registro para
1175  FC03 32F5FB       STA     Unallocated$Record$Count;  la primera "escritura"
1176  FC06 3C           INR     A                       ;Indicación de lectura real que debe
1177  FC07 32F8FB       STA     Read$Operation          ;  realizarse forzando una prelectura
1178  FC0A 32F7FB       STA     Must$Preread$Sector     ;  del sector para situarlo en la
1179                                                    ;  memoria intermedia de disco
1180  FC0D 3E02         MVI     A,Write$Unallocated     ;Simulación de desbloqueo en respuesta
1181  FC0F 32E3FB       STA     Write$Type              ;  a si se trata de la primera escritura
1182                                                    ;  sobre un bloque no asignado.
1183  FC12 C36EFC       JMP     Perform$Read$Write      ;Utilización del prog. común para ejecutar la lec.
```

Figura 6-4. (Continuación.)

```
1184                    ;
1185                    ;  Escritura de un sector de 128 bytes desde la dirección DMA en curso
1186                    ;  en el disco, pista y sector seleccionados previamente.
1187                    ;
1188                    ;  Llegado este punto, el BDOS habrá puesto a uno el registro C para indicar
1189                    ;  que esta operación de escritura es sobre un bloque ya asignado
1190                    ;  (debido a que puede haberse necesitado la prelectura del sector),
1191                    ;  al directorio (caso en que los datos deben escribirse directamente
1192                    ;  en el disco), o en el primer sector de 128 bytes de un bloque
1193                    ;  desasignado previamente.
1194                    ;
1195                    ;  Unicamente la escritura del directorio tiene lugar inmediatamente. En
1196                    ;  los demás casos, los datos se transferirán desde la dirección DMA a la memoria
1197                    ;  intermedia del disco y sólo escritas realmente en el momento en que las
1198                    ;  circunstancias fuerzan la transferencia. El número de operaciones con el
1199                    ;  disco físico pueden reducirse considerablemente de esta forma.
1200                    ;
1201                    WRITE:
1202  FC15 3AF9FB           LDA     Deblocking$Required     ;Comprobación de si es necesario el desbloqueo
1203  FC18 B7               ORA     A                       ;(indicador puesto a uno en la llamada a SELDSK)
1204  FC19 CA4DFD           JZ      Write$No$Deblock
1205
1206  FC1C AF               XRA     A                       ;Indicación de que se precisa una operación de
1207  FC1D 32F8FB           STA     Read$Operation          ;  escritura (es decir, NO de lectura)
1208  FC20 79               MOV     A,C                     ;Guardar tipo de escritura del BDOS
1209  FC21 32E3FB           STA     Write$Type
1210  FC24 FE02             CPI     Write$Unallocated       ;Comprobación de si se trata de una primera
1211                                                        ;  escritura de un bloque no asignado
1212  FC26 C237FC           JNZ     Check$Unallocated$Block ;No, comprobación de si está escribiendo
1213                                                        ;  en un bloque
1214                                                        ;Sí, escritura primero de un bloque
1215                                                        ;  no asignado -- inicialización
1216                                                        ;  de las variables asociadas
1217                                                        ;  con escritura desasignada
1218  FC29 3E10             MVI     A,Allocation$Block$Size/128     ;Obtención del número
1219                                                                ;  de sectores de 128 bytes
1220  FC2B 32F5FB           STA     Unallocated$Record$Count        ;  y activación del contador
1221                                                                ;
1222  FC2E 21EAFB           LXI     H,Selected$Dk$Trk$Sec           ;Copia de disco, pista y sector
1223  FC31 11F1FB           LXI     D,Unallocated$Dk$Trk$Sec        ;  en variables desasignadas
1224  FC34 CD35FD           CALL    Move$Dk$Trk$Sec
1225                    ;
1226                    ;  Comprobación de si se trata de la primera escritura
1227                    ;  en un bloque no asignado -- si lo es, el contador
1228                    ;  de registro no asignados ha sido puesto en el número
1229                    ;  de sectores de 128 bytes del bloque.
1230                    ;
1231                    Check$Unallocated$Block:
1232  FC37 3AF5FB           LDA     Unallocated$Record$Count
1233  FC3A B7               ORA     A
1234  FC3B CA66FC           JZ      Request$Preread         ;No, es una escritura sobre
1235                                                        ;  un bloque asignado
1236                                                        ;Sí, es una escritura sobre
1237                                                        ;  un bloque no asignado
1238  FC3E 3D               DCR     A                       ;Decrementar el número de sectores 128 bytes
1239                                                        ;  no escritos en el bloque
1240  FC3F 32F5FB           STA     Unallocated$Record$Count        ;  y almacenar el nuevo valor
1241
1242  FC42 21EAFB           LXI     H,Selected$Dk$Trk$Sec   ;Comprobación de si el disco, pista y
1243  FC45 11F1FB           LXI     D,Unallocated$Dk$Trk$Sec;  sector seleccionados son los mismos que
1244  FC48 CD29FD           CALL    Compare$Dk$Trk$Sec      ;  los del bloque no asignado
1245  FC4B C266FC           JNZ     Request$Preread         ;No, es necesaria la prelectura
1246                                                        ;Sí, no se precisa prelectura
1247                                                        ;Este momento es apropiado para actualizar el
1248                                                        ;  sector en curso y para mirar si es necesario
1249                                                        ;  también actualizar la pista
1250                    ;
1251                                                        ;Por diseño, Compare$Dk$Trk$Sec
1252                                                        ; retorna con
1253                                                        ; DE -> Unallocated$Sector
1254  FC4E EB               XCHG                          ; HL -> Unallocated$Sector
1255  FC4F 34               INR     M                       ;Actualización Unallocated$Sector
1256  FC50 7E               MOV     A,M                     ;Comprobación de si ahora sector > máximo
1257  FC51 FE48             CPI     CPM$Sec$Per$Track       ; en la pista
1258  FC53 DA5FFC           JC      No$Track$Change         ;No (A < M)
1259                                                        ;Sí
```

Figura 6-4. (Continuación.)

```
1260  FC56 3600          MVI     M,0                     ;Puesta a cero del sector
1261  FC58 2AF2FB        LHLD    Unallocated$Track       ;Aumentar en 1 la pista
1262  FC5B 23            INX     H
1263  FC5C 22F2FB        SHLD    Unallocated$Track
1264                     ;
1265                     No$Track$Change:
1266                                                     ;Indicación de hasta qué código no es
1267                                                     ;  necesaria la prelectura
1268  FC5F AF            XRA     A
1269  FC60 32F7FB        STA     Must$Preread$Sector     ;Must$Preread$Sector=0
1270  FC63 C36EFC        JMP     Perform$Read$Write
1271                     ;
1272                     Request$Preread:
1273  FC66 AF            XRA     A                       ;Indicación de que no se trata de una escritura
1274  FC67 32F5FB        STA     Unallocated$Record$Count        ;  de un bloque no asignado
1275  FC6A 3C            INR     A
1276  FC6B 32F7FB        STA     Must$Preread$Sector     ;Indicación de que es necesaria la
1277                                                     ;  prelectura del sector físico
1278                     ;
1279                     ;
1280                     Perform$Read$Write:             ;Programa común para ejecutar lecturas y
1281                                                     ;  escrituras de sectores de 128 bytes
1282  FC6E AF            XRA     A                       ;Suposición de que no habrán
1283  FC6F 32F6FB        STA     Disk$Error$Flag         ;  errores de disco
1284
1285  FC72 3AEDFB        LDA     Selected$Sector         ;Conversión del sector seleccionado de 128 bytes
1286  FC75 1F            RAR                             ;  en sector físico dividiendo por 4
1287  FC76 1F            RAR
1288  FC77 E63F          ANI     3FH                     ;Borrado de los bits no deseados
1289  FC79 32EEFB        STA     Selected$Physical$Sector
1290                                                     ;
1291  FC7C 21E8FB        LXI     H,Data$In$Disk$Buffer   ;Comprobación de si la memoria intermedia de
1292  FC7F 7E            MOV     A,M                     ;  disco contiene datos
1293  FC80 3601          MVI     M,1                     ;(Condición no verdadera indica que la
1294                                                     ; memoria intermedia contiene datos)
1295  FC82 B7            ORA     A                       ;¿Es cierto que contiene datos?
1296  FC83 CAA3FC        JZ      Read$Sector$into$Buffer ;No, se leen datos del sector físico
1297                                                     ;  en la memoria intermedia
1298
1299                                                 ;La memoria intermedia contiene
1300                                                 ;  un sector físico
1301                                                 ;  Nota: El disco, pista y sector físico
1302                                                 ;  en la memoria intermedia deben
1303                                                 ;  ser comprobados, de ahí
1304                                                 ;  el uso de la subrutina Compare$Dk$Trk
1305                                                 ;
1306  FC86 11E4FB        LXI     D,In$Buffer$Dk$Trk$Sec  ;Comprobación de si el sector en la mem. int. es
1307  FC89 21EAFB        LXI     H,Selected$Dk$Trk$Sec   ;  el mismo que el que ha sido seleccionado
1308  FC8C CD24FD        CALL    Compare$Dk$Trk          ;Comparación de disco y pista únicamente
1309  FC8F C29CFC        JNZ     Sector$Not$In$Buffer    ;No, debe ser leído
1310
1311  FC92 3AE7FB        LDA     In$Buffer$Sector        ;Obtención sector físico en la memoria intermedia
1312  FC95 21EEFB        LXI     H,Selected$Physical$Sector
1313  FC98 BE            CMP     M                       ;Comprobación de si sector físico correcto
1314  FC99 CAB1FC        JZ      Sector$In$Buffer        ;Sí, está todavía en la memoria
1315                     ;
1316                     Sector$Not$In$Buffer:
1317                                                     ;No, deberá ser leído en los contenidos
1318                                                     ;  actuales de la memoria intermedia
1319  FC9C 3AE9FB        LDA     Must$Write$Buffer       ;Comprobación de si la memoria intermedia contiene
1320  FC9F B7            ORA     A                       ;  datos que deben ser escritos antes
1321  FCA0 C495FD        CNZ     Write$Physical          ;Sí, escritura
1322                     ;
1323                     Read$Sector$into$Buffer:
1324  FCA3 CD11FD        CALL    Set$In$Buffer$Dk$Trk$Sec        ;Fijación de disco, pista
1325                                                     ; y sector en las variables de la memoria
1326                                                     ;  intermedia para reflejar cuál es el sector
1327                                                     ;  actualmente en la memoria intermedia
1328  FCA6 3AF7FB        LDA     Must$Preread$Sector     ;En la práctica el sector necesita únicamente
1329  FCA9 B7            ORA     A                       ;  ser leído físicamente si se precisa
1330                                                     ;  la prelectura
1331  FCAA C49AFD        CNZ     Read$Physical           ;Sí, prelectura del sector
1332  FCAD AF            XRA     A                       ;Puesta a cero de la marca para reflejar el
1333  FCAE 32E9FB        STA     Must$Write$Buffer       ;  contenido de la memoria intermedia
1334                     ;
1335                     Sector$In$Buffer:           ;El sector seleccionado en la pista y disco correctos
```

Figura 6-4. (Continuación.)

```
1336                                            ;  está todavía en la memoria intermedia.
1337                                            ;Conversión del sector CP/M (128 bytes)
1338                                            ;  seleccionado en dirección relativa baja
1339                                            ;  en la memoria intermedia
1340  FCB1 3AEDFB       LDA     Selected$Sector ;Obtención número de sector seleccionado
1341  FCB4 E603         ANI     Sector$Mask     ;Enmascarar sólo los bits menos significativos
1342  FCB6 6F           MOV     L,A             ;Multiplicación por 128 desplazando un valor de 16 bits
1343  FCB7 2600         MVI     H,0             ;  siete posiciones a la izquierda
1344  FCB9 29           DAD     H               ;* 2
1345  FCBA 29           DAD     H               ;* 4
1346  FCBB 29           DAD     H               ;* 8
1347  FCBC 29           DAD     H               ;* 16
1348  FCBD 29           DAD     H               ;* 32
1349  FCBE 29           DAD     H               ;* 64
1350  FCBF 29           DAD     H               ;* 128
1351                ;
1352  FCC0 1133F6       LXI     D,Disk$Buffer   ;Obtención dirección base de la memoria intermedia de disco
1353  FCC3 19           DAD     D               ;Añadir a número de sector * 128
1354                                            ;HL -> dirección de comienzo del sector de 128 bytes en la
1355                                            ;  memoria intermedia de disco
1356  FCC4 EB           XCHG                    ;DE -> sector en la memoria intermedia de disco
1357  FCC5 2A63FB       LHLD    DMA$Address     ;Obtención de la dirección DMA fijada en la llamada a SETDMA
1358  FCC8 EB           XCHG                    ;Suposición de operación de lectura; por tanto:
1359                                            ;  DE -> dirección DMA
1360                                            ;  HL -> sector en la memoria intermedia de disco
1361  FCC9 0E10         MVI     C,128/8         ;Debido a la rapidez del método utilizado para transferir
1362                                            ;  datos dentro y fuera de la memoria intermedia de disco
1363                                            ;  (8 bytes transferidos por bucle de iteración)
1364                                            ;  el contador necesita ser sólo de un octavo
1365                                            ;  de lo normal
1366                                            ;En este punto -
1367                                            ;       C = contador de lazos
1368                                            ;       DE -> dirección DMA
1369                                            ;       HL -> sectores en la memoria intermedia de disco
1370  FCCB 3AF8FB       LDA     Read$Operation  ;Determinar en qué caso los datos son transferidos
1371  FCCE B7           ORA     A               ;  fuera de la memoria intermedia (lectura)
1372  FCCF C2D7FC       JNZ     Buffer$Move     ;  o al interior (escritura)
1373                                            ;Escritura en la memoria intermedia
1374                                                    ;(A debe ser 0)
1375  FCD2 3C           INR     A                       ;Fijar marca para forzar la escritura de la
1376  FCD3 32E9FB       STA     Must$Write$Buffer       ;  memoria intermedia de disco
1377  FCD6 EB           XCHG                            ;Hacer DE -> sector en memoria intermedia de disco
1378                                                    ;       HL -> dirección DMA
1379                ;
1380                ;
1381                Buffer$Move:                ;El bucle de transferencias siguiente transfiere 8 bytes
1382                                            ;  a la vez de HL a DE, C contiene el contador
1383                                            ;  de bucle
1384  FCD7 7E           MOV     A,M             ;Obtención de byte de la fuente
1385  FCD8 12           STAX    D               ;Puesta en el destino
1386  FCD9 13           INX     D               ;Actualización de apuntadores
1387  FCDA 23           INX     H
1388  FCDB 7E           MOV     A,M             ;Obtención de byte de la fuente
1389  FCDC 12           STAX    D               ;Puesta en el destino
1390  FCDD 13           INX     D               ;Actualización de apuntadores
1391  FCDE 23           INX     H
1392  FCDF 7E           MOV     A,M             ;Obtención de byte de la fuente
1393  FCE0 12           STAX    D               ;Puesta en el destino
1394  FCE1 13           INX     D               ;Actualización de apuntadores
1395  FCE2 23           INX     H
1396  FCE3 7E           MOV     A,M             ;Obtención de byte de la fuente
1397  FCE4 12           STAX    D               ;Puesta en el destino
1398  FCE5 13           INX     D               ;Actualización de apuntadores
1399  FCE6 23           INX     H
1400  FCE7 7E           MOV     A,M             ;Obtención de byte de la fuente
1401  FCE8 12           STAX    D               ;Puesta en el destino
1402  FCE9 13           INX     D               ;Actualización de apuntadores
1403  FCEA 23           INX     H
1404  FCEB 7E           MOV     A,M             ;Obtención de byte de la fuente
1405  FCEC 12           STAX    D               ;Puesta en el destino
1406  FCED 13           INX     D               ;Actualización de apuntadores
1407  FCEE 23           INX     H
1408  FCEF 7E           MOV     A,M             ;Obtención de byte de la fuente
1409  FCF0 12           STAX    D               ;Puesta en el destino
1410  FCF1 13           INX     D               ;Actualización de apuntadores
```

Figura 6-4. (Continuación.)

```
1411  FCF2 23          INX     H
1412  FCF3 7E          MOV     A,M             ;Obtención byte de la fuente
1413  FCF4 12          STAX    D               ;Puesta en destino
1414  FCF5 13          INX     D               ;Actualización de apuntadores
1415  FCF6 23          INX     H
1416
1417  FCF7 0D          DCR     C               ;Decrementar contador de bucle
1418  FCF8 C2D7FC      JNZ     Buffer$Move     ;Repetir hasta sector CP/M transferido
1419                                           ;
1420  FCFB 3AE3FB      LDA     Write$Type      ;Si escritura en directorio, escriba inmediatamente
1421  FCFE FE01        CPI     Write$Directory ;  la memoria intermedia
1422  FD00 3AF6FB      LDA     Disk$Error$Flag ;Obtención marca de error en caso de retraso en lec. o esc.
1423  FD03 C0          RNZ                     ;Retorno si retraso en lectura o escritura
1424                                           ;
1425  FD04 B7          ORA     A               ;Comprobación de si se han producido errores de disco
1426  FD05 C0          RNZ                     ;Sí, abandono de intento de escritura del directorio
1427                                           ;
1428  FD06 AF          XRA     A               ;Borrar marca que indica que la memoria intermedia
1429  FD07 32E9FB      STA     Must$Write$Buffer       ;debe ser escrita en disco
1430  FD0A CD95FD      CALL    Write$Physical  ;Escritura de la memoria intermedia en el sector físico
1431  FD0D 3AF6FB      LDA     Disk$Error$Flag ;Retorno de indicador a la rutina que efectuó la llamada
1432  FD10 C9          RET
1433                ;
1434                ;
1435                Set$In$Buffer$Dk$Trk$Sec:          ;Indicación de qué disco seleccionado, pista y
1436                                                   ;  sector residen en la memoria intermedia
1437  FD11 3AEAFB      LDA     Selected$Disk
1438  FD14 32E4FB      STA     In$Buffer$Disk
1439
1440  FD17 2AEBFB      LHLD    Selected$Track
1441  FD1A 22E5FB      SHLD    In$Buffer$Track
1442
1443  FD1D 3AEEFB      LDA     Selected$Physical$Sector
1444  FD20 32E7FB      STA     In$Buffer$Sector
1445
1446  FD23 C9          RET
1447                ;
1448                Compare$Dk$Trk:                    ;Comparación de disco y pista apuntados
1449                                                   ;  por DE y HL
1450  FD24 0E03        MVI     C,3                     ;Disco (1), pista (2)
1451  FD26 C32BFD      JMP     Compare$Dk$Trk$Sec$Loop ;Utilización de programa común
1452
1453                Compare$Dk$Trk$Sec:                ;Comparación de variables de disco, pista y sector
1454                                                   ;  apuntados por DE y HL
1455  FD29 0E04        MVI     C,4                     ;Disco (1), pista (2) y sector (1)
1456                Compare$Dk$Trk$Sec$Loop:
1457  FD2B 1A          LDAX    D                       ;Obtención comparador
1458  FD2C BE          CMP     M                       ;Comparación con comparando
1459  FD2D C0          RNZ                             ;Abandono de la comparación si se encuentra desigualdad
1460  FD2E 13          INX     D                       ;Actualización apuntador comparado
1461  FD2F 23          INX     H                       ;Actualización apuntador comparando
1462  FD30 0D          DCR     C                       ;Decrementar contador de bucle
1463  FD31 C8          RZ                              ;Retorno (con indicador de cero a uno)
1464  FD32 C32BFD      JMP     Compare$Dk$Trk$Sec$Loop
1465                ;
1466                ;
1467                Move$Dk$Trk$Sec:                   ;Transferencia de las variables
1468                                                   ;  apuntadas por HL a las apuntadas
1469                                                   ;  por DE
1470  FD35 0E04        MVI     C,4                     ;Disco (1), pista (2) y sector (1)
1471                Move$Dk$Trk$Sec$Loop:
1472  FD37 7E          MOV     A,M                     ;Obtención byte fuente
1473  FD38 12          STAX    D                       ;Almacenamiento en destino
1474  FD39 13          INX     D                       ;Actualización apuntadores
1475  FD3A 23          INX     H
1476  FD3B 0D          DCR     C                       ;Decrementar contador de bytes
1477  FD3C C8          RZ                              ;Retorno si transferidos todos los bytes
1478  FD3D C337FD      JMP     Move$Dk$Trk$Sec$Loop
1479                ;
1480                ;
1482                ;
1483                ;  Existen dos controladores de disco "inteligentes"
1484                ;  en este sistema, uno para disquetes de 8'' y otro para unidades
1485                ;  de 5 1/4".
1486                ;
1487                ;  Los controladores están "conectados físicamente" al monitor en ciertos lugares
```

Figura 6-4. (Continuación.)

```
1488                    ;   en la memoria para detectar cuándo debe realizarse alguna
1489                    ;   operación de disco. Monitor-controlador de 8'' posición 0040H.
1490                    ;   Monitor-controlador de 5 1/4'' posición 0045H. Son los
1491                    ;   llamados bytes de control de disco. Si el bit más significativo
1492                    ;   del byte de control de disco está a 1, el controlador examinará
1493                    ;   la palabra que sigue a los bytes de control respectivos.
1494                    ;   Esta palabra debe contener la dirección de una tabla de control
1495                    ;   de disco válida que especifique la operación exacta a realizar.
1496                    ;
1497                    ;   Una vez que la operación ha terminado, el controlador pone
1498                    ;   su byte de control a 00H, lo que indica al programa que la tarea
1499                    ;   se ha completado.
1500                    ;
1501                    ;   El controlador activa también un código de retorno en el bloque de estado
1502                    ;   de disco -- ambos controladores utilizan para ello la misma posición 0043H.
1503                    ;   Si el primer byte de este bloque de estado es menor que 80H, significa que
1504                    ;   ha ocurrido un error. No hay más detalles relevantes respecto a la fijación
1505                    ;   de estados de este BIOS simple. Nótese que el controlador de disco tiene
1506                    ;   construidos -- en lógica de varios intentos -- lecturas y escritura que se
1507                    ;   intentan diez veces antes de que el controlador dé un error.
1508                    ;
1509                    ;   La disposición de la tabla de control de disco se muestra a continuación.
1510                    ;   Nótese que los controladores tienen la capacidad de encadenar tablas de
1511                    ;   control de manera que pueden lanzarse operaciones de disco en cadena. Esta
1512                    ;   característica no se usa en este BIOS. El controlador precisa, sin embargo,
1513                    ;   que los apuntadores de cadena en la tabla de control de disco sean
1514                    ;   apuntadas por los bytes de control principal para indicar el final de la
1515                    ;   cadena.
1516                    ;
1517  0040 =            Disk$Control$8                EQU     40H     ;Byte de control de 8"
1518  0041 =            Command$Block$8               EQU     41H     ;Apuntador a tabla de control
1519                    ;
1520  0043 =            Disk$Status$Block             EQU     43H     ;Bloque de estado de 8" y de 5 1/4"
1521                    ;
1522  0045 =            Disk$Control$5                EQU     45H     ;Byte de control de 5 1/4"
1523  0046 =            Command$Block$5               EQU     46H     ;Apuntador a tabla de control
1524                    ;
1525                    ;
1526                    ;   Floppy Disk Control Tables
1527                    ;
1528  FD40 00           Floppy$Command:               DB      0       ;Orden
1529  0001 =            Floppy$Read$Code              EQU     01H
1530  0002 =            Floppy$Write$Code             EQU     02H
1531  FD41 00           Floppy$Unit:                  DB      0       ;Unidad número 0 ó 1
1532  FD42 00           Floppy$Head:                  DB      0       ;Cabeza número 0 ó 1
1533  FD43 00           Floppy$Track:                 DB      0       ;Número de pista
1534  FD44 00           Floppy$Sector:                DB      0       ;Número de sector
1535  FD45 0000         Floppy$Byte$Count:            DW      0       ;Número de bytes a leer/escribir
1536  FD47 0000         Floppy$DMA$Address:           DW      0       ;Dirección de transferencia
1537  FD49 0000         Floppy$Next$Status$Block:     DW      0       ;Apuntador a bloque de estado siguiente
1538                                                                  ;   si existen órdenes encadenadas
1539  FD4B 0000         Floppy$Next$Control$Location: DW      0       ;Apuntador a byte de control siguiente
1540                                                                  ;   si existen órdenes encadenadas
1541                    ;
1542                    ;
1543                    ;
1544                    Write$No$Deblock:                             ;Escritura del contenido de la memoria intermedia
1545                                                                  ;   en el sector correcto
1546  FD4D 3E02             MVI     A,Floppy$Write$Code             ;Obtención de código de función de escritura
1547  FD4F C354FD           JMP     Common$No$Deblock               ;Ir a programa común
1548                    Read$No$Deblock:                              ;Lectura previa del sector seleccionado en la
1549                                                                  ;   memoria intermedia
1550  FD52 3E01             MVI     A,Floppy$Read$Code              ;Obtención código de función de lectura
1551                    Common$No$Deblock:
1552  FD54 3240FD           STA     Floppy$Command          ;Activación código de función de orden
1553                                                        ;Activación tabla de órdenes no desbloqueadas
1554  FD57 218000           LXI     H,128                   ;Bytes por sector
1555  FD5A 2245FD           SHLD    Floppy$Byte$Count
1556  FD5D AF               XRA     A                       ;El disquete de 8" tiene únicamente la cabeza 0
1557  FD5E 3242FD           STA     Floppy$Head
1558                                                        ;
1559  FD61 3AEAFB           LDA     Selected$Disk           ;El controlador de disquetes de 8" sólo tiene información
1560                                                        ;   en unidades 0 y 1 de manera que Selected$Disk
1561                                                        ;   debe convertirse
1562  FD64 E601             ANI     01H                     ;Transformación a 1 ó 0
1563  FD66 3241FD           STA     Floppy$Unit             ;Fijar número de unidad
```

Figura 6-4. (Continuación.)

```
1564                                        ;
1565  FD69 3AEBFB       LDA     Selected$Track
1566  FD6C 3243FD       STA     Floppy$Track      ;Fijar número de pista
1567                                        ;
1568  FD6F 3AEDFB       LDA     Selected$Sector
1569  FD72 3244FD       STA     Floppy$Sector     ;Fijar número de sector
1570                                        ;
1571  FD75 2A63FB       LHLD    DMA$Address       ;Transferencia directa entre dirección DMA
1572  FD78 2247FD       SHLD    Floppy$DMA$Address      ;y controlador de 8"
1573                                        ;
1574                                        ;El controlador de disco acepta tablas de control de disco
1575                                        ;  encadenadas, pero en este caso no se utilizan, de manera
1576                                        ;  que los apuntadores al "siguiente" deben apuntar
1577                                        ;  a los bytes de control inicial en la página
1578                                        ;  base
1579  FD7B 214300       LXI     H,Disk$Status$Block       ;Apuntar estado siguiente al bloque
1580  FD7E 2249FD       SHLD    Floppy$Next$Status$Block  ;  de estado principal
1581                                                      ;
1582  FD81 214000       LXI     H,Disk$Control$8          ;Apuntar byte de control del siguiente
1583  FD84 224BFD       SHLD    Floppy$Next$Control$Location  ;  al byte de control principal
1584                                                      ;
1585  FD87 2140FD       LXI     H,Floppy$Command          ;Apuntar controlador a tabla de control
1586  FD8A 224100       SHLD    Command$Block$8
1587                                                      ;
1588  FD8D 214000       LXI     H,Disk$Control$8          ;Activar controlador para realizar
1589  FD90 3680         MVI     M,80H                     ;  la operación
1590  FD92 C3F7FD       JMP     Wait$For$Disk$Complete
1591
1592                ;
1593                ;
1594
1595                Write$Physical:                 ;Escritura del contenido de la memoria intermedia
1596                                                ;  en el sector correcto
1597  FD95 3E02         MVI     A,Floppy$Write$Code   ;Obtención de código de función de escritura
1598  FD97 C39CFD       JMP     Common$Physical       ;Ir a un código común
1599                Read$Physical:                  ;Lectura previa del sector seleccionado
1600                                                ;  en la memoria intermedia
1601  FD9A 3E01         MVI     A,Floppy$Read$Code    ;Obtención código de función de lectura
1602                ;
1603                Common$Physical:
1604  FD9C 3240FD       STA     Floppy$Command        ;Activación tabla de órdenes
1605
1606                ;
1607  FD9F 3AFAFB       LDA     Disk$Type             ;Obtención tipo de disco (fijado en SELDSK)
1608  FDA2 FE01         CPI     Floppy$5              ;Confirmación de disquete de 5 1/4"
1609  FDA4 CAADFD       JZ      Correct$Disk$Type     ;Sí
1610  FDA7 3E01         MVI     A,1                   ;No, inidicar error de disco
1611  FDA9 32F6FB       STA     Disk$Error$Flag
1612  FDAC C9           RET
1613                Correct$Disk$Type:              ;Activación tabla de control de disco
1614                                                ;
1615  FDAD 3AE4FB       LDA     In$Buffer$Disk        ;Convertir número de disco a 0 ó 1
1616  FDB0 E601         ANI     1                     ;  para controlador de disco
1617  FDB2 3241FD       STA     Floppy$Unit
1618
1619  FDB5 2AE5FB       LHLD    In$Buffer$Track       ;Activar número de pista
1620  FDB8 7D           MOV     A,L                   ;Nota: Es un valor de un solo byte
1621  FDB9 3243FD       STA     Floppy$Track          ; para el controlador
1622                                                  ;
1623                                                  ;El sector debe convertirse en un número
1624                                                  ;  de cabeza y un número de sector.
1625                                                  ;  Sectores 0 - 8 tienen cabeza 0, 9 - 17
1626                                                  ;  tienen cabeza 1
1627  FDBC 0600         MVI     B,0                   ;Suposición de cabeza 0
1628  FDBE 3AE7FB       LDA     In$Buffer$Sector      ;Obtención número de sector físico
1629  FDC1 4F           MOV     C,A                   ;Almacenar copia en caso de ser cabeza 0
1630  FDC2 FE09         CPI     9                     ;Comprobación de si < 9
1631  FDC4 DACBFD       JC      Head$0                ;Si, < 9
1632  FDC7 D609         SUI     9                     ;No, modificar número de sector anterior
1633                                                  ;  en el rango 0 a 8
1634  FDC9 4F           MOV     C,A                   ;Puesta del sector en B
1635  FDCA 04           INR     B                     ;Fijación a cabeza 1
1636                Head$0:
1637  FDCB 78           MOV     A,B                   ;Fijación número de cabeza
1638  FDCC 3242FD       STA     Floppy$Head
1639  FDCF 79           MOV     A,C                   ;Fijación número de sector
```

Figura 6-4. (Continuación.)

```
1640  FDD0 3C          INR     A                       ;  (los sectores físicos comienzan en 1)
1641  FDD1 3244FD      STA     Floppy$Sector
1642                                                   ;
1643  FDD4 210002      LXI     H,Physical$Sector$Size  ;Fijar contador de bytes
1644  FDD7 2245FD      SHLD    Floppy$Byte$Count
1645                                                   ;
1646  FDDA 2133F6      LXI     H,Disk$Buffer           ;Fijación de dirección de transferencia a
1647  FDDD 2247FD      SHLD    Floppy$DMA$Address      ;  memoria intermedia de disco
1648                                                   ;
1649                                                   ;Si se utiliza únicamente una tabla de control,
1650                                                   ;  se cierra el estado y se hace apuntar
1651                                                   ;  los apuntadores de cadena a los bytes
1652                                                   ;  de control principal
1653  FDE0 214300      LXI     H,Disk$Status$Block
1654  FDE3 2249FD      SHLD    Floppy$Next$Status$Block
1655  FDE6 214500      LXI     H,Disk$Control$5
1656  FDE9 224BFD      SHLD    Floppy$Next$Control$Location
1657
1658  FDEC 2140FD      LXI     H,Floppy$Command        ;Activar apuntador a bloque de comandos
1659  FDEF 224600      SHLD    Command$Block$5
1660
1661  FDF2 214500      LXI     H,Disk$Control$5        ;Activar controlador de disco de 5 1/4"
1662  FDF5 3680        MVI     M,80H
1663                ;
1664                Wait$For$Disk$Complete:            ;Espera hasta que el bloque de estado de disco
1665                                                   ;  indique que se ha completado la operación,
1666                                                   ;  entonces comprobación de ocurrencia de errores.
1667                                                   ;En entrada HL -> byte de control de disco
1668  FDF7 7E          MOV     A,M                     ;Obtención byte de control
1669  FDF8 B7          ORA     A
1670  FDF9 C2F7FD      JNZ     Wait$For$Disk$Complete  ;Operación no realizada todavía
1671
1672  FDFC 3A4300      LDA     Disk$Status$Block       ;Terminada -- comprobar estado ahora
1673  FDFF FE80        CPI     80H                     ;Comprobación de ocurrencia de errores
1674  FE01 DA09FE      JC      Disk$Error              ;Sí
1675  FE04 AF          XRA     A                       ;No
1676  FE05 32F6FB      STA     Disk$Error$Flag         ;Borrado de indicador de error
1677  FE08 C9          RET
1678                Disk$Error:
1679  FE09 3E01        MVI     A,1                     ;Fijación de indicador de error de disco a
1680  FE0B 32F6FB      STA     Disk$Error$Flag         ;  distinto-de-cero
1681  FE0E C9          RET
1682                ;
1683                ;
1684                ;
1685                ;  Imagen de la tabla de control de disco para arranque caliente
1686                ;
1687                Boot$Control$Part$1:
1688  FE0F 01          DB      1                       ;Función de lectura
1689  FE10 00          DB      0                       ;Número de unidad
1690  FE11 00          DB      0                       ;Número de cabeza
1691  FE12 00          DB      0                       ;Número de pista
1692  FE13 02          DB      2                       ;Número de sector de comienzo
1693  FE14 0010        DW      8*512                   ;Número de bytes a leer
1694  FE16 00E0        DW      CCP$Entry               ;Lectura en esta dirección
1695  FE18 4300        DW      Disk$Status$Block       ;Apuntador a bloque de estado siguiente
1696  FE1A 4500        DW      Disk$Control$5          ;Apuntador a tabla de control siguiente
1697                Boot$Control$Part2:
1698  FE1C 01          DB      1                       ;Función de lectura
1699  FE1D 00          DB      0                       ;Número de unidad
1700  FE1E 01          DB      1                       ;Número de cabeza
1701  FE1F 00          DB      0                       ;Número de pista
1702  FE20 01          DB      1                       ;Número de sector de comienzo
1703  FE21 0006        DW      3*512                   ;Número de bytes a leer
1704  FE23 00F0        DW      CCP$Entry + (8*512)     ;Lectura en esta dirección
1705  FE25 4300        DW      Disk$Status$Block       ;Apuntador a bloque de estado siguiente
1706  FE27 4500        DW      Disk$Control$5          ;Apuntador a tabla de control siguiente
1707
1708                ;
1709                ;
1710                ;
1711                WBOOT:              ;Entrada de arranque caliente
1712                                    ;Cuando se da un arranque caliente, el CCP y el BDOS
1713                                    ;  deben recargarse en la memoria. En este BIOS sólo se
1714                                    ;  utilizarán disquetes de 5 1/4", por lo que su programa
```

Figura 6-4. (Continuación.)

```
1715                                    ;  es hardware específico del controlador. Se utilizan
1716                                    ;  dos tablas de control prefabricadas
1717  FE29 318000        LXI     SP,80H
1718  FE2C 110FFE        LXI     D,Boot$Control$Part1    ;Ejecutar la primera lectura de arranque caliente
1719  FE2F CD3BFE        CALL    Warm$Boot$Read          ;Cargar unidad 0, pista 0, cabeza 0,
1720                                                     ;  sectores 2 a 8
1721  FE32 111CFE        LXI     D,Boot$Control$Part2    ;Ejecutar segunda lectura
1722  FE35 CD3BFE        CALL    Warm$Boot$Read          ;Cargar unidad 0, pista 0, cabeza 1,
1723                                                     ;  sectores 1 a 3
1724  FE38 C340F8        JMP     Enter$CPM               ;Activación de página base y entrada del CCP
1725                  ;
1726                  Warm$Boot$Read:                   ;En entrada, DE -> imagen de tabla de control
1727                                                    ;Esta tabla de control se transfiere a la tabla
1728                                                    ;  de control de disco principal y se activa
1729                                                    ;  entonces el controlador
1730  FE3B 2140FD        LXI     H,Floppy$Command        ;HL -> tabla de control real
1731  FE3E 224600        SHLD    Command$Block$5         ;Indicar al controlador su dirección
1732                                                    ;Transferir la imagen de la tabla de control
1733                                                    ;  a la propia tabla de control
1734  FE41 0E0D          MVI     C,13                    ;Activar contador de byte
1735                  Warm$Boot$Move:
1736  FE43 1A            LDAX    D                       ;Obtención de byte imagen
1737  FE44 77            MOV     M,A                     ;Almacenar en la tabla de control real
1738  FE45 23            INX     H                       ;Actualización de apuntadores
1739  FE46 13            INX     D
1740  FE47 0D            DCR     C                       ;Decrementar contador de bytes
1741  FE48 C243FE        JNZ     Warm$Boot$Move          ;Continúe hasta transferencia de todos los bytes
1742
1743  FE4B 214500        LXI     H,Disk$Control$5        ;Activar controlador
1744  FE4E 3680          MVI     M,80H
1745                  Wait$For$Boot$Complete:
1746  FE50 7E            MOV     A,M                     ;Obtención byte de estado
1747  FE51 B7            ORA     A                       ;Comprobación de completada
1748  FE52 C250FE        JNZ     Wait$For$Boot$Complete  ;No
1749                                                     ;Sí, comprobación de errores
1750  FE55 3A4300        LDA     Disk$Status$Block
1751  FE58 FE80          CPI     80H
1752  FE5A DA5EFE        JC      Warm$Boot$Error         ;Sí, ocurrencia de error
1753  FE5D C9            RET
1754                  ;
1755                  Warm$Boot$Error:
1756  FE5E 2167FE        LXI     H,Warm$Boot$Error$Message
1757  FE61 CD33F8        CALL    Display$Message
1758  FE64 C329FE        JMP     WBOOT                   ;Iniciar de nuevo arranque caliente
1759                  ;
1760                  Warm$Boot$Error$Message:
1761  FE67 0D0A576172    DB      CR,LF,'Warm Boot Error - retrying...',CR,LF,0
1762                  ;
1763                  ;
1764  FE89               END     ;Final del listado del BIOS
```

Figura 6-4. (Continuación.)

Pasos principales
Construcción del primer sistema
Utilización del SYSGEN para escribir el CP/M en el disco
Utilización del DDT para construir la imagen del CP/M en memoria
El cargador de autocarga (bootstrap loader) del CP/M
Utilización de MOVCPM para reubicar el CCP y el BDOS
Colocación del conjunto

7

Construcción de un nuevo sistema CP/M

Este capítulo describe cómo construir una versión del CP/M con su propio BIOS incorporado. Muestra también cómo poner el CP/M en un disco flexible y cómo escribir un cargador para llevar el CP/M a la memoria.

El fabricante de la computadora juega un papel significativo en la construcción de un nuevo sistema CP/M. Algunos de los programas de utilidad del CP/M pueden modificarse por los fabricantes para adaptarlos a los sistemas individuales. Desgraciadamente no todos los fabricantes hacen estos programas a medida. Por tanto, debería invertirse algún tiempo en estudiar la documentación que se suministra con el sistema, con el fin de ver hasta qué punto se ha hecho. También deberán ensamblarse e imprimirse listados de todos los ficheros en lenguaje ensamblador de su disquete CP/M.

Es imposible predecir los detalles de esta personalización y los procedimientos especiales que los fabricantes pueden instalar en cada sistema particular. Por consiguiente, este capítulo describe en primer lugar el conjunto del mecanismo de construcción de un sistema CP/M, y en segundo

lugar los detalles de la construcción de un sistema CP/M alrededor del ejemplo de BIOS mostrado en el capítulo anterior como figura 6-4.

Pasos principales

La construcción de un nuevo sistema CP/M consiste, en grandes líneas, en los siguientes pasos:

- Creación de un BIOS nuevo o modificado con los controladores de dispositivos adecuados en él. Ensamblar esto de forma que ejecute al final de la memoria [mediante el uso de una relación de *origen* (ORG) para fijar el contador de posiciones].
- Creación de nuevas versiones del CCP y el BDOS con todas las direcciones de las instrucciones cambiadas de forma que se sitúen correctamente en la memoria inmediatamente bajo el nuevo BIOS. Digital Research proporciona una utilidad especial denominada MOVCPM para realizar esta tarea.
- Creación o modificación de un cargador del CP/M que será cargado por el firmware que trabaja cuando se conecta por primera vez la computadora (o se presiona el botón de RESET). Normalmente, el cargador del CP/M se ejecuta en el final de la dirección inferior de la memoria. La dirección exacta y los detalles de cualquier inicialización de hardware que deba realizar dependerán enteramente de la computadora particular.
- Utilizando los programas de utilidad estándar de Digital Research, situar el cargador, el CCP y el BDOS y el BIOS juntos en la parte inferior de la memoria. Entonces escribir esta nueva versión de CP/M en un disco en los sitios adecuados. De nuevo, dependiendo del diseño de la computadora, es posible utilizar el programa de utilidad estándar, SYSGEN, para escribir la imagen completa de CP/M en el disco. En caso contrario, se debe escribir un programa especial que haga esto.

Cuando el CP/M está funcionando todavía en la computadora y se quieren añadir nuevas posibilidades al BIOS, todo lo que se necesita es cambiar el BIOS y volver a construir el sistema. El CCP y el BDOS necesitarán ser transportados abajo en la memoria si el cambio expande el BIOS de manera significativa. Si esto ocurre, habrá que hacer cambios menores en el cargador, de forma que lea la nueva imagen del CP/M en la memoria en una dirección más baja y transfiera el control al sitio correcto (la primera instrucción del vector de saltos del BIOS).

Construcción del primer sistema

La primera vez que se implemente el CP/M, una buena idea es no realizar ningún cambio en el BIOS. Simplemente ensamblar el código fuente del BIOS y proceder a construir el sistema. Entonces, si el nuevo sistema no funciona, se sabe que existe algún fallo en el procedimiento que se está usando antes de que se hayan introducido nuevas prestaciones o modificaciones del código fuente del BIOS. Los cambios en el BIOS pueden oscurecer fácilmente cualquier problema que surja en el propio procedimiento de construcción.

Los ingredientes

Para construir el CP/M se necesitarán los siguientes ficheros y programas de utilidad:

- El código fuente en lenguaje ensamblador del BIOS. Habrá que buscar en el disquete CP/M un fichero con un nombre como CBIOS.ASM (Sistema de entrada/salida básico personalizado). Algunos fabricantes no suministran el código fuente de sus BIOS; lo venden separadamente o simplemente no lo modifican. Si no se puede tener el código fuente, el único camino a seguir para poder añadir nuevas características al BIOS es escribiendo enteramente el BIOS desde el principio.
- El código fuente para el cargador del CP/M. Este también deberá encontrarse en el disquete o disponible separadamente en el fabricante de la computadora.
- El ensamblador de Digital Research, que convierte el código fuente en un lenguaje máquina de forma hexadecimal. Este programa, denominado ASM.COM, se hallará en el disquete CP/M. Los ensambladores equivalentes, tales como los macro-ensambladores MAC y RMAC de Digital Research o el Microsoft M80, también pueden ser utilizados.
- La utilidad de Digital Research denominada MOVCPM, que prepara una imagen en memoria del CCP y el BDOS con todas las direcciones ajustadas a los valores exactos.
- Los programas de depuración de Digital Research, denominados DDT (Dynamic Debugging Tool), o la versión mejorada para el chip CPU Z80, ZSID (Z80 Symbolic Interactive Debugger). El DDT se utiliza para leer en los distintos ficheros de programas y formar una imagen en memoria del sistema CP/M.

- El programa de utilidad de Digital Research SYSGEN. Este escribe la imagen de memoria compuesta por el cargador, CCP, BDOS y BIOS en el disco. SYSGEN se diseñó para trabajar en sistemas de disco flexible. Si la computadora utiliza un disco rígido, puede tenerse un programa con un nombre tal como PUTCPM o WRITECPM que realiza la misma función.

La finalidad última

En la figura 6-4, líneas 0044 a 0065, se pueden ver las equivalencias que definen las direcciones de base para el CCP, el BDOS y el BIOS. La figura 7-1 muestra cómo aparecerá el principio de la memoria cuando esta versión del CP/M se ha cargado en memoria.

Todo sería más fácil si se pudiera construir esta imagen en memoria en las direcciones mostradas y escribir la imagen fuera en el disco. La construcción de esta imagen, sin embargo, probablemente escribirá sobre la versión del CP/M con que se está operando porque éste también se encuentra al principio de la memoria. Por consiguiente, la finalidad es crear una réplica de esta imagen más abajo en la memoria, pero con todas las direcciones de instrucción fijas para ser *ejecutadas* en las direcciones que muestra la figura 7-1.

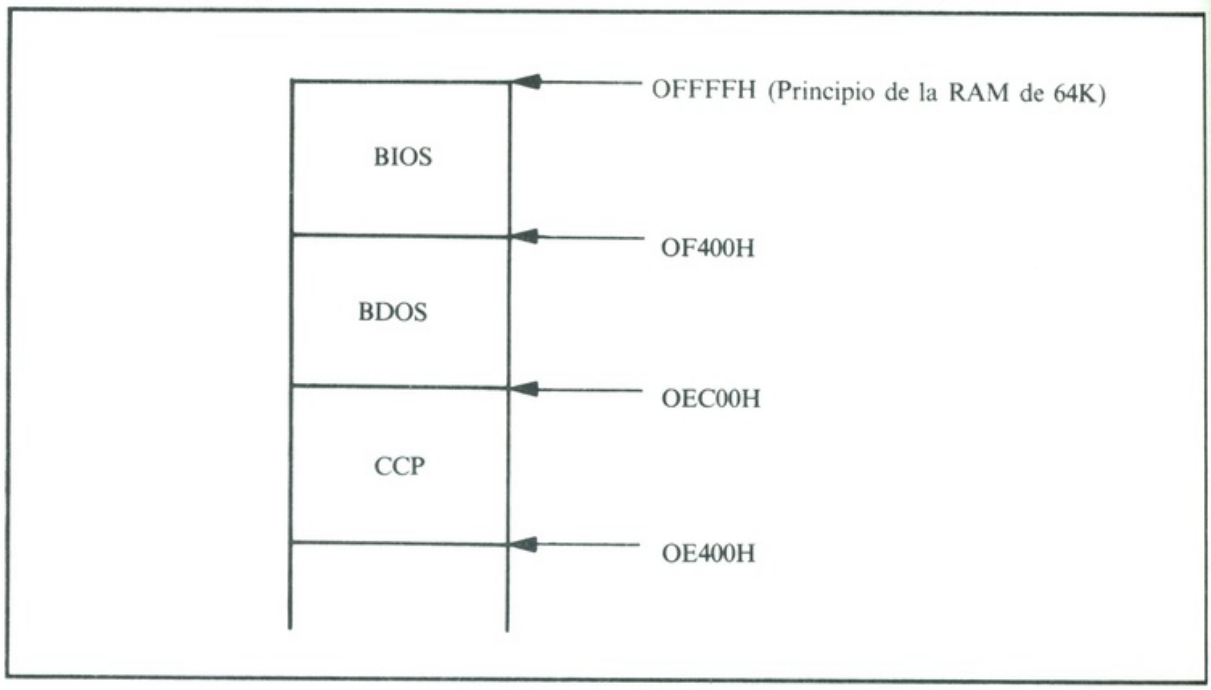


Figura 7-1. Distribución de la memoria del CP/M.

Utilización del SYSGEN para escribir el CP/M en el disco

La utilidad SYSGEN escribe una imagen de memoria en un disco lógico específico. Puede usar una imagen de memoria que se prepara para que esté en la memoria antes de invocar a SYSGEN, o se puede dirigir el SYSGEN para que lea en un fichero de disco que contiene la imagen. También se puede usar SYSGEN para transportar un sistema CP/M existente desde un disquete a otro dirigiéndolo para que cargue la imagen del CP/M desde un disquete en la memoria y escribir entonces esa imagen en otro disquete.

Lea detenidamente la documentación suministrada por el fabricante de la computadora para asegurarse de que puede utilizarse SYSGEN en el sistema. El SYSGEN, como se edita por Digital Research, está construido para funcionar con disquetes de 8 pulgadas, una sola cara y simple densidad. Si el sistema no utiliza estos disquetes estándar, SYSGEN deberá ser adaptado al sistema de disco que se utilice.

Cuando SYSGEN carga una imagen del CP/M en la memoria, colocará el cargador, el CCP, el BDOS y el BIOS en las direcciones previamente determinadas que se muestran en la figura 7-2, sin tener en cuenta dónde se ha originado este CP/M.

Como puede verse, la colocación *relativa* entre los componentes no ha cambiado; la imagen total simplemente se ha desplazado hacia abajo en la memoria, por debajo de la versión del CP/M en curso de ejecución. El arranque ha añadido al cuadro exactamente debajo del CCP.

La utilidad SYSGEN escribe esta imagen en un disquete empezando en el sector 1 de la pista 0 y continuando hasta el sector 26 de la pista 1. Consúltese la figura 2-2 para ver la distribución del CP/M en un disquete normalizado de 8 pulgadas, una sola cara y simple densidad.

Si se requiere al SYSGEN para que lea la imagen de memoria desde un fichero (lo que se realiza llamando el SYSGEN con el nombre de fichero en la misma línea que la llamada al SYSGEN), entonces SYSGEN supone que se ha creado previamente en la memoria la imagen correcta y se la ha salvado (con la orden SAVE). SYSGEN omite entonces los 16 primeros sectores del fichero, como si evitara escribir sobre sí mismo.

A continuación hay un ejemplo de cómo usar SYSGEN para transferir la imagen del CP/M de un disquete a otro:

```
A>SYSGEN<CR>
SYSGEN VER 2.0
SOURCE DRIVE NAME (OR RETURN TO SKIP) A
SOURCE ON A:, THEN TYPE RETURN <cr>
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
DESTINATION ON B: THEN TYPE RETURN <cr>
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) <cr>
A>_
```

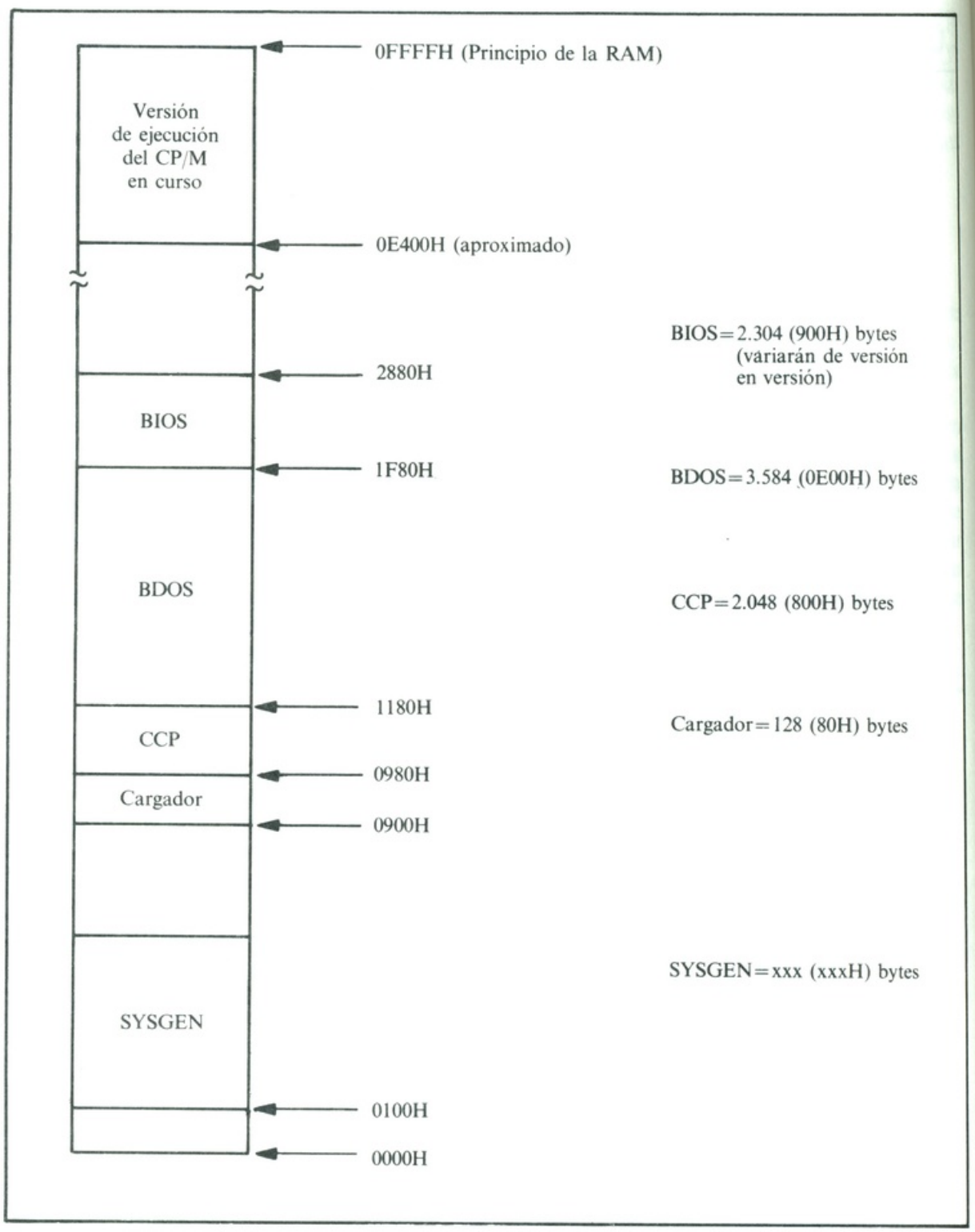


Figura 7-2. Distribución de la memoria del SYSGEN.

Como puede verse, SYSGEN da la oportunidad de especificar el nombre de la unidad fuente o mecanografiar CARRIAGE RETURN. Si se introduce un CARRIAGE RETURN, SYSGEN supone que la imagen del CP/M ya está en memoria. Obsérvese que se necesita llamar al SYSGEN sólo una vez para escribir la misma imagen del CP/M en más de un disco.

Un BIOS mayor que el estándar puede producir dificultades al usar SYSGEN. El formato SYSGEN estándar sólo permite contener el BIOS en seis sectores de 128 bytes, por lo que si éste es mayor de 768 (300H) bytes, será un problema. La imagen del CP/M no se ajusta a las dos primeras pistas de un disquete estándar de 8 pulgadas.

Actualmente es difícil encontrar un sistema de disquetes donde haya que cargar el CP/M desde un disquete de una sola cara y simple densidad. La mayoría de los sistemas utilizan ahora disquetes de doble cara y doble densidad como formato normal, pero pueden cambiar a disquetes de una cara y simple densidad para intercambiar información con otras computadoras.

Como no hay ningún formato "estándar" para disquetes de 8 pulgadas, doble cara y doble densidad, probablemente no se podrán leer los escritos en sistemas de una marca o un modelo distintos. Por consiguiente, es necesario únicamente estar enterados de la forma de usar una distribución de disco que conserve los discos compatibles con otras máquinas que sean exactamente las mismas que las que se poseen.

Esto también es verdad si se tienen disquetes de 5 1/4 pulgadas. No existe un estándar industrial para éstos, por lo que la consideración principal será colocar el directorio de ficheros en el mismo lugar en que estará en los escritos por otros usuarios del mismo modelo de computadora. También hay que asegurarse de usar el mismo entrelazamiento de sectores. En caso contrario, se obtendrá una versión deformada siempre que se intente leer ficheros originarios de otros sistemas.

Con los disquetes de mayor capacidad, se puede reservar más espacio para conservar la imagen del CP/M en el disquete. Por ejemplo, en el caso del BIOS que se muestra en la figura 6-4, la imagen del CP/M se escribe en uno de 5 1/4 pulgadas, de doble cara y doble densidad, utilizando sectores de 512 bytes. La figura 7-3 muestra la distribución de este disquete. Obsérvese que el cargador está situado en un sector de 512 bytes por sí mismo. Lo que hace mucho más simple el programa de autocarga y de arranque caliente del BIOS.

La imagen de memoria deberá alterarse para reflejar el hecho de que el cargador ocupa ahora un sector completo de 512 bytes. Mejor que cambiar todas las direcciones, el cargador se carga en la memoria 384 (180H) bytes más abajo, de forma que acaba en la misma dirección que antes. La figura 7-4 muestra la imagen de memoria revisada.

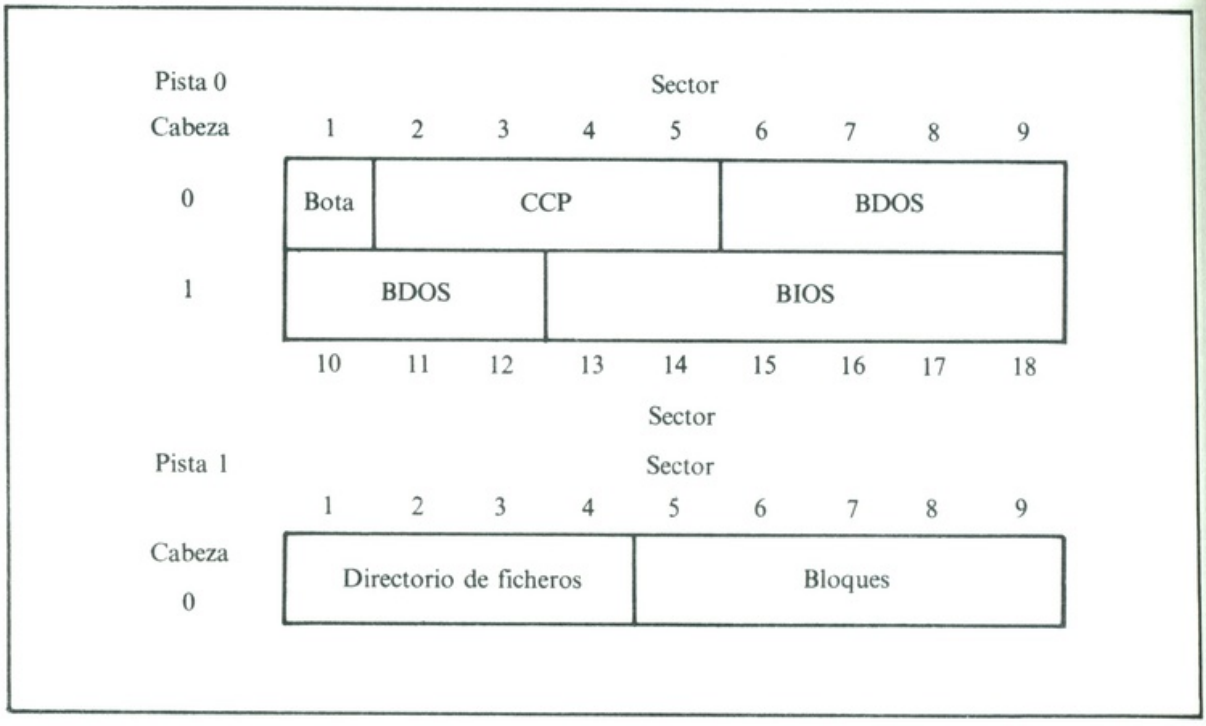


Figura 7-3. Distribución de disco para ejemplificar el BIOS en disquetes de 5 1/4 pulgadas.

Escritura de una utilidad PUTCPM

Como el sistema del ejemplo usa disquetes de 5 1/4 pulgadas con sectores de 512 bytes, la versión estándar de SYSGEN no puede ser utilizada para escribir la imagen del CP/M en un disquete. Habrá de usarse un reemplazamiento funcional suministrado por el fabricante de la computadora o desarrollar un pequeño programa de utilidad para hacer este trabajo.

La figura 7-5 muestra un ejemplo de tal programa. Está escrito con un propósito general, de forma que se pueda utilizar para cualquier sistema cambiando las equivalencias del principio del programa de forma que reflejen las especificaciones de las unidades de disco.

Obsérvese que existen dos problemas para resolver. Primero, el área del disco en el cual reside la imagen del CP/M no es accesible para el BDOS, porque se encuentra fuera del área del sistema de ficheros del disco. Segundo, es raro escribir la imagen del CP/M en el disco sin ninguna clase de entrelazamiento entre sectores, ya que al hacerlo así se retrasaría el proceso de carga. En algún caso, el entrelazamiento sería redundante, porque el cargador no está realizando ningún proceso distinto al de lectura del disco y, por tanto, puede leer éste sin entrelazamiento.

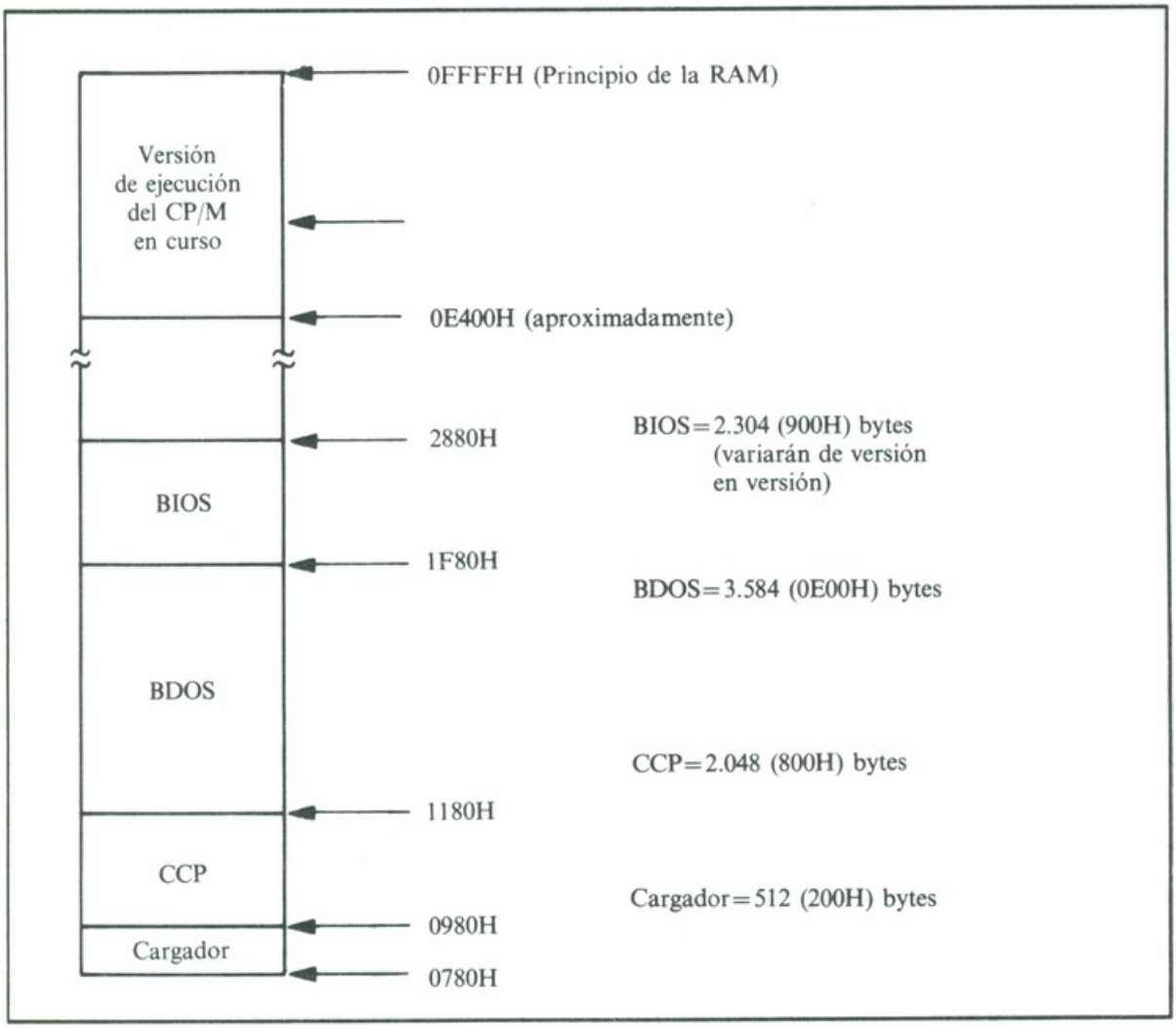


Figura 7-4. Direcciones para ejemplificación de la imagen del BIOS.

Utilización del DDT para construir la imagen del CP/M en memoria

El DDT, programa de depuración de Digital Research, se utiliza para leer ficheros del tipo ".COM" y ".HEX" en la memoria. Entender la estructura interna de estos tipos de fichero es importante, tanto en lo que se refiere a lo que el DDT puede hacer, como en lo que se refiere a la forma en que MOVCPM puede cambiar efectivamente un fichero en código máquina de forma que pueda ser ejecutado en una nueva dirección de la memoria.

```
                ;       Este programa escribe en un disco flexible el cargador,
                ;       el CCP, el BDOS y el BIOS. Corre bajo CP/M como
                ;       un programa normal.
                ;
3130 =          Version         EQU     '01'    ;Equivalencias utilizadas en el mensaje
                                                ; de "funcionando"
3730 =          Month           EQU     '07'
3432 =          Day             EQU     '24'
3238 =          Year            EQU     '82'
                ;
                ;
                ;       El verdadero programa PUTCPMF5 consiste en este programa,
                ;       más los BOOTF5.HEX, CCP, BDOS y BIOS.
                ;
                ;       Cuando se ejecuta este programa, la memoria imagen
                ;       verá lo siguiente:
                ;
                ;             Componente        Dirección base
                ;               BIOS              1F80H
                ;               BDOS              1180H
                ;               CCP               0980H
                ;               BOOTF5            0780H
                ;
                ;       Los programas se presentan como sigue:
                ;
                ;               BIOS.HEX          Por ensamblar el código fuente
                ;               BDOS )            Desde un fichero CPMnn.COM por MOVCPM
                ;               CCP  )              y almacenado en disco
                ;               BOOTF5.HEX        Por ensamblaje del código fuente
                ;
                ;       Los componentes se sitúan juntos utilizando DDT con las
                ;       siguientes órdenes:
                ;
                ;               DDT CPMnn.COM
                ;               IPUTCPMF5.HEX
                ;               R                           (Lee en este programa)
                ;               IBOOTF5.HEX
                ;               R680                        (Lee en BOOT en 0780H)
                ;               IBIOS.HEX
                ;               R2980                       (Lee en BIOS 1F80H)
                ;               GO                          (Salida de DDT)
                ;               SAVE 40 PUTCPMF5.COM        (Creación del fichero definitivo .COM)
                ;
                ;       La distribución real en el disquete es como sigue:
                ;
                ; Pista 0                       Sector
                ;          1     2     3     4     5     6     7     8     9
                ; Cabeza +-----+-----+-----+-----+-----+-----+-----+-----+-----+
                ;   0    |Boot |<======== CCP ========>|<=======  BDOS ========|
                ;        +-----+-----+-----+-----+-----+-----+-----+-----+-----+
                ;   1    |======= BDOS ====>|<============  BIOS  ============>|
                ;        +-----+-----+-----+-----+-----+-----+-----+-----+-----+
                ;          10    11    12    13    14    15    16    17    18
                ;                              Sector
                ;
                ;       Equivalencias para definir el tamaño de la memoria y la dirección base y
                ;       la longitud de los componentes del sistema
                ;
0040 =          Memory$Size     EQU     64      ;Número de Kbytes de RAM
                ;
                ;       La longitud del BIOS debe coincidir con la declarada en el BIOS
                ;
0900 =          BIOS$Length     EQU     0900H
                ;
0200 =          Boot$Length     EQU     512
0800 =          CCP$Length      EQU     0800H   ;Constante
0E00 =          BDOS$Length     EQU     0E00H   ;Constante
                ;
1F00 =          Length$In$Bytes EQU     CCP$Length + BDOS$Length + BIOS$Length
                ;
0780 =          Start$Image     EQU     980H - Boot$Length      ;Dirección de la imagen CP/M
2100 =          Length$Image    EQU     Length$In$Bytes + Boot$Length
                ;
                ;
```

Figura 7-5. Ejemplo de PUTCPM.

```
                        ;       Características del disco
                        ;
                        ;       Estas equivalencias definen las características físicas
                        ;       del disquete de manera que el programa pueda ir de un sector
                        ;       al siguiente actualizando la pista y poniendo a cero
                        ;       el sector cuando sea necesario.
                        ;
0001 =                  First$Sector$on$Track   EQU     1
0012 =                  Last$Sector$on$Track    EQU     18
0009 =                  Last$Sector$on$Head$0   EQU     9
0200 =                  Sector$Size             EQU     512
                        ;
                        ;
                        ;       Características del controlador
                        ;
                        ;       En esta computadora el controlador puede escribir
                        ;       varios sectores con una sola orden. Sin embargo,
                        ;       para dar un mejor ejemplo, se muestra sólo
                        ;       la lectura de un sector cada vez.
                        ;
0001 =                  Sectors$Per$Write       EQU     1
                        ;
                        ;
                        ;       Cold boot characteristics
                        ;
0000 =                  Start$Track             EQU     0       ;Valores iniciales de la imagen CP/M
0001 =                  Start$Sector            EQU     1       ;= " =
0011 =                  Sectors$To$Write        EQU     (Length$Image + Sector$Size - 1) / Sector$Size
                        ;


                        ;
0009 =                  B$PRINTS        EQU     9       ;Impresión de cadena terminada por $
0005 =                  BDOS            EQU     5       ;Punto de entrada al BDOS
                        ;
                        ;

0100                            ORG     100H
                        Put$CPM:
0100 C33F01                     JMP     Main$Code       ;Entrada del cuerpo del programa principal
                                                        ;Para claridad del programa, las estructuras
                                                        ;  de datos se muestran antes del código
                                                        ;  ejecutable
000D =                  CR      EQU     0DH             ;Retorno de carro
000A =                  LF      EQU     0AH             ;Salto de línea
                        ;
                        Signon$Message:
0103 0D0A507574                 DB      CR,LF,'Put CP/M on Diskette'
0119 0D0A                       DB      CR,LF
011B 5665727369                 DB      'Version '
0123 3031                       DW      Version
0125 20                         DB      ' '
0126 3037                       DW      Month
0128 2F                         DB      '/'
0129 3234                       DW      Day
012B 2F                         DB      '/'
012C 3832                       DW      Year
012E 0D0A24                     DB      CR,LF,'$'

                        ;
                        ;        Disk control tables
                        ;
0045 =                  Disk$Control$5  EQU     45H     ;Byte de control 5 1/4"
0046 =                  Command$Block$5 EQU     46H     ;Apuntador a tabla de control
0043 =                  Disk$Status     EQU     43H     ;Terminación del estado
                        ;
                        ;
                        ;       La pista de tablas de órdenes y la DMA$Address pueden utilizarse
                        ;       como memoria de trabajo y actualizadas cuando continúa el
                        ;       proceso de carga. El sector en la tabla de órdenes no puede
                        ;       utilizarse directamente como lo necesita el controlador de
                        ;       disco, debe ser el número de sector sobre la cabeza especificada
                        ;       (1 -- 9) y más que el número de sector en la pista. Debe usarse,
                        ;       por tanto, una variable separada.
                        ;
```

Figura 7-5. (Continuación.)

```
0131 01         Sector:         DB      Start$Sector
                ;
0132 02         Command$Table:  DB      02H             ;Orden -- escritura
0133 00         Unit:           DB      0               ;Unidad 0 ó 1
0134 00         Head:           DB      0               ;Número de cabeza 0 ó 1
0135 00         Track:          DB      Start$Track     ;Utilizada como variable de trabajo
0136 00         Sector$on$head: DB      0               ;Convertido a controlador de bajo nivel
0137 0002       Byte$Count:     DW      Sector$Size * Sectors$Per$Write
0139 8007       DMA$Address:    DW      Start$Image
013B 4300       Next$Status:    DW      Disk$Status     ;Apuntador a bloque de estado siguiente
                                                        ;  si las órdenes están encadenadas
013D 4500       Next$Control:   DW      Disk$Control$5  ;Apuntador al siguiente byte de control
                                                        ;  si las órdenes están encadenadas

                Main$Code:
013F 310001             LXI     SP,Put$CPM      ;La pila crece hacia abajo debajo del programa

0142 110301             LXI     D,Signon$Message        ;Funcionando
0145 0E09               MVI     C,B$PRINTS              ;Imprimir cadena hasta $
0147 CD0500             CALL    BDOS

014A 213201             LXI     H,Command$Table         ;Apuntar el controlador de disco

014D 224600             SHLD    Command$Block$5         ;  al bloque de comandos

0150 0E11               MVI     C,Sectors$To$Write      ;Fijar contador de sectores
                Write$Loop:
0152 CD7C01             CALL    Put$CPM$Write           ;Escritura de datos en el disquete
0155 0D                 DCR     C                       ;Decrementar contador
0156 CA0000             JZ      0                       ;Arranque caliente

0159 213101             LXI     H,Sector                ;Actualización número de sector
015C 3E01               MVI     A,Sectors$Per$Write     ;  añadiendo el número de sectores
015E 86                 ADD     M                       ;  al controlador
015F 77                 MOV     M,A                     ;Guardar resultado
0160 3E13               MVI     A,Last$Sector$On$Track + 1      ;Comprobación de final de pista
0162 BE                 CMP     M
0163 C26F01             JNZ     Not$End$Track

0166 3601               MVI     M,First$Sector$On$Track ;Sí, puesta al comienzo
0168 2A3501             LHLD    Track                   ;Actualización número de pista
016B 23                 INX     H
016C 223501             SHLD    Track

                Not$End$Track:
016F 2A3901             LHLD    DMA$Address             ;Actualización dirección de DMA
0172 110002             LXI     D,Sector$Size * Sectors$Per$Write
0175 19                 DAD     D
0176 223901             SHLD    DMA$Address
0179 C35201             JMP     Write$Loop              ;Escritura del bloque siguiente
                ;
                Put$CPM$Write:                  ;En este punto la descripción de la operación
                                                ;  que se requiere está en las variables contenidas
                                                ;  en la tabla de órdenes a lo largo
                                                ;  de la variable de sector

017C C5                 PUSH    B                       ;Guardar contador de sector en C

                ;------ Cambiar esta rutina para coincidir con el controlador de disco utilizado ------

017D 0600               MVI     B,0                     ;Suposición de cabeza 0
017F 3A3101             LDA     Sector                  ;Obtención de sector requerido
0182 4F                 MOV     C,A                     ;Obtención de copia de él
0183 FE0A               CPI     Last$Sector$on$Head$0+1 ;Comprobación de cabeza 1
0185 DA8C01             JC      Head$0                  ;No
0188 D609               SUI     Last$Sector$on$Head$0   ;Para cabeza 1
018A 4F                 MOV     C,A                     ;Guardar copia
018B 04                 INR     B                       ;Activar cabeza 1
                Head$0:
018C 78                 MOV     A,B                     ;Obtención cabeza
018D 323401             STA     Head
0190 79                 MOV     A,C                     ;Obtención sector
0191 323601             STA     Sector$On$Head
```

Figura 7-5. (Continuación.)

```
0194 214500             LXI     H,Disk$Control$5         ;Activación del controlador
0197 3680               MVI     M,80H

                Wait$For$Boot$Complete:
0199 7E                 MOV     A,M                      ;Obtención byte de estado
019A B7                 ORA     A                        ;Comprobación de completo
019B C29901             JNZ     Wait$For$Boot$Complete   ;No
                                                         ;Si, comprobación de errores
019E 3A4300             LDA     Disk$Status
01A1 FE80               CPI     80H
01A3 DAA801             JC      Put$CPM$Error            ;Si, ha ocurrido un error

                ;------ Fin de la rutina de escritura ------

01A6 C1                 POP     B                        ;Recuperar contador de sectores en C
01A7 C9                 RET
                ;
                Put$CPM$Error:
01A8 11B301             LXI     D,Put$CPM$Error$Message
01AB 0E09               MVI     C,B$PRINTS               ;Impresión de cadena hasta $
01AD CD0500             CALL    BDOS                     ;Salida de mensaje de error
01B0 C33F01             JMP     Main$Code                ;Reinicializar cargador
                ;

                Put$CPM$Error$Message:
01B3 0D0A457272         DB      CR,LF,'Error in writing CP/M - retrying...',CR,LF,'$'
01DB                    END     Put$CPM
```

Figura 7-5. (Continuación.)

Estructura de ficheros ".COM"

Un fichero COM es una imagen de memoria. Es una réplica de las configuraciones de bits que se van a crear cuando el fichero es cargado en la memoria. Los ficheros COM se diseñan normalmente para cargar a partir de la situación 100H. Ninguna estructura interna del fichero lo requiere; por tanto, si se conocen los contenidos, no hay nada que impida cargarlo en la memoria empezando en alguna dirección distinta de 100H.

Como puede observarse en la descripción del CP/M en el capítulo 4, la orden SAVE construida en el CCP permite crear un fichero COM especificando el número de "páginas" de memoria de 256 bytes y el nombre del mismo. El CCP escribirá una imagen exacta de la memoria a partir de la situación 100H.

Estructura de ficheros ".HEX"

Los ficheros HEX son generados por el ensamblador. Contienen una representación de carácter ASCII de valores hexadecimales. Por ejemplo, los contenidos de un solo byte de memoria con el valor binario 10101111 estarían representados por dos caracteres ASCII, A F, en un fichero HEX.

El fichero HEX posee un nivel estructural más alto que una serie de

caracteres ASCII. Cada línea de caracteres ASCII se acaba por un CARRIAGE RETURN/LINE FEED. La estructura completa se muestra en la figura 7-6.

El aspecto más importante de un fichero HEX es que cada línea contiene la dirección en la cual se cargan los bytes de datos. Cada línea se procesa independientemente, por lo que las direcciones de carga de líneas sucesivas no necesitan estar en orden.

El DDT puede leer en un fichero HEX en una dirección diferente de aquella en que el programa ha de estar preparado para llevar a cabo la ejecución. Por ejemplo, se puede leer el fichero HEX del BIOS en el sitio correspondiente a la imagen de memoria (mostrada en la figura 7-4). Hay dos formas de utilizar el DDT para leer en un fichero COM o HEX. Se puede especificar el nombre del fichero en la misma línea de orden con el DDT. Por ejemplo:

```
A>DDT B:XYZ.HEX<cr>          <- Llamada al DDT con nombre
DDT VERS 2.0                 <- de fichero
NEXT  PC
0180 0100                    <- ... y visualiza el siguiente
                                byte libre y dirección del punto de entrada
                             <- ... y demanda una orden
```

La ventaja de este método de cargar un fichero es que se puede especificar el disco lógico en que hay que buscar el fichero. El segundo modo de utilizar el DDT es cargar primero éste y entonces, cuando ya ha lanzado su mensaje, especificar el nombre de fichero y solicitar que el DDT lo cargue en la forma siguiente:

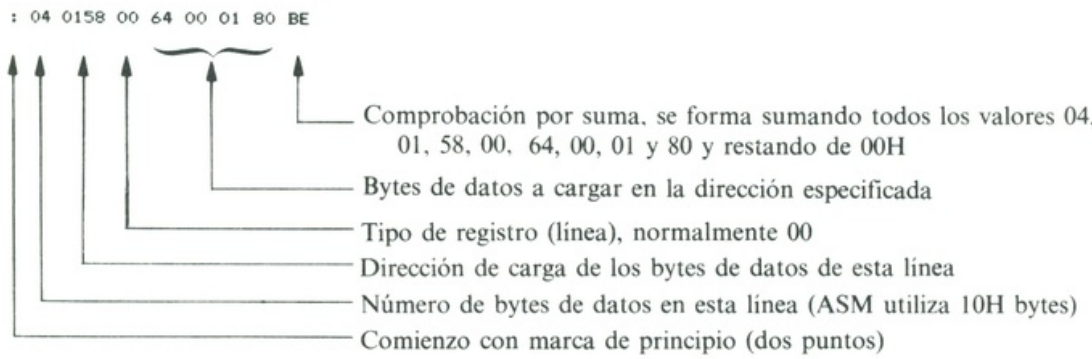
```
-Ifilename.typ<cr>           <- Entrar nombre y tipo de fichero
-R<cr>                       <- Lectura en el fichero
```

La orden "I" inicializa el bloque de control del fichero por defecto en la página base (en la situación 005CH) con el nombre y el tipo de fichero; *no* fija el disco lógico. Si se necesita esto último, primero hay que fijar manualmente el primer byte del FCB por defecto en la siguiente forma:

```
-Ifilename.typ<cr>           <- Especificación del nombre de fichero
-S5C<cr>                     <- "S",(SET) fijar posición 5C
005C 00 02<cr>               <- Era 00, se entra 02<cr>
005D 41 .<cr>                <- Entrada de "." para terminar
-R<cr>                       <- Lectura en el fichero
```

La posición 005CH deberá ponerse a 01H para la unidad A, a 02H para B, etc.

La orden "R" leerá en los ficheros HEX para las direcciones de *ejecución* especificadas en cada línea de fichero HEX, por lo que habrá de tener precaución —si se olvida poner una sentencia ORG (origen) al principio del código fuente en lenguaje ensamblador, leyendo en el fichero HEX resultan-



NOTA: Los ficheros HEX no contienen blancos; en el ejemplo anterior se han dejado espacios vacíos únicamente por claridad.

Figura 7-6. Ejemplo de línea de fichero HEX.

te se sobrepasará la situación 0000H, destruyendo los contenidos de la página base—. De forma similar, si se intentase leer en el fichero HEX para un BIOS, existe una gran probabilidad de escribir destructivamente sobre el sistema CP/M actualmente en ejecución.

El DDT toma el tipo de fichero que se introduce como parte del nombre. Para tipos de fichero distintos de HEX, el DDT carga el fichero empezando en la situación 0100H.

La orden "R" puede utilizarse también para leer ficheros en memoria en diferentes direcciones. Esto se lleva a cabo mecanografiando un número hexadecimal inmediatamente después de R, sin ninguna puntuación. Para los ficheros HEX, el número que se introduce se añade a la dirección en cada línea del fichero HEX y la suma se utiliza como dirección en la cual se cargan los bytes de datos. Los bytes de datos no cambian sino sólo la dirección de carga.

Para ficheros COM, el número que se introduce se añade a 0100H y la suma se utiliza como dirección de inicio para cargar el fichero.

La suma se realiza como aritmética de 16 bits sin signo, ignorando cualquier acarreo, de forma que se puede cargar un fichero HEX del BIOS en la memoria inferior utilizando la orden "R", con lo que se denomina "valor de desplazamiento".

Si un fichero HEX se ha ensamblado para ejecutarlo en la dirección "ejec.", y se necesita usar el DDT para leer en este fichero en la dirección "carga", es preciso resolver la siguiente ecuación:

Desplazamiento = carga − ejec.

La orden "H" del DDT configura la aritmética hexadecimal. Calcula y presenta la suma y la diferencia entre dos valores hexadecimales. Por ejemplo, el BIOS de la figura 6-4 ha sido ensamblado para *ejecutarse* en la posición 0F600H, pero necesita ser *cargado* en la memoria en la posición 1F80H. A continuación se indica la forma de calcular el desplazamiento correcto para la orden "R":

```
-H1F80,F600<cr>          <- Utilizar orden H
1580,2980                <- Suma, diferencia
```

Así pues, para leer en el fichero HEX del BIOS denominado FIG6-4.HEX en la posición 1F80H, habría que introducir las siguientes órdenes en el DDT:

```
-IFIG6-4.HEX<cr>         <- Especifica nombre y tipo de fichero
-R2980<cr>               <- Carga en 0F600H + 2980H (= 1F80H)
```

De esta forma, utilizando el DDT, se puede leer en los ficheros HEX tanto para el BIOS como para el cargador.

El cargador de autocarga (bootstrap loader) del CP/M

El cargador es llevado a la memoria por el firmware basado en una PROM en la computadora. Carga el CCP, BDOS y BIOS y transfiere entonces el control al punto de entrada del arranque en frío del BIOS —la primera instrucción de salto del vector de saltos del BIOS.

El cargador es un programa aislado; no puede hacer uso de las funciones del CP/M porque ninguna de éstas se encuentra en memoria cuando se necesita el cargador. Este firmware en la PROM que ha cargado el cargador puede contener algunas subrutinas que pueden utilizarse por el propio cargador, pero esto variará de un sistema a otro.

La figura 7-7 muestra el programa del cargador para el BIOS ejemplificado (de la figura 6-4). Este programa se ha escrito de forma general para poder adaptarlo a cualquier sistema. El controlador de disco del sistema del ejemplo, de hecho, puede leer en múltiples sectores del disco, pero generalmente el código que se muestra lee únicamente un sector al tiempo, lo que aumenta considerablemente el tiempo requerido para cargar el CP/M, pero hace más general el cargador.

Obsérvese que casi lo primero que hace el cargador es enviar a la consola un mensaje de conexión. Esto confirma no sólo el número de versión, sino que demuestra que el cargador se ha cargado satisfactoriamente.

El código residente en PROM se ha diseñado para cargar el cargador del CP/M en la posición 100H, permitiendo que el programa sea depurado

```
                        ;       Ejemplo de cargador CP/M
                        ;
                        ;       Este programa está escrito desde la pista 0, cabeza 0, sector 1
                        ;       por el programa PUTCPMF5.
                        ;       Es cargado en la memoria a partir de la posición 100H por el mecanismo
                        ;       de carga basado en PROM que obtiene el control del procesador
                        ;       al conectarlo o en una puesta a cero del sistema.
                        ;
3130 =                  Version         EQU     '01'    ;Equivalencias usadas en el mensaje de "funcionando"
3730 =                  Month           EQU     '07'
3432 =                  Day             EQU     '24'
3238 =                  Year            EQU     '82'
                        ;
0000 =                  Debug           EQU     0       ;Fijar a distinto de cero para depurar como un
                                                        ;  programa transitorio normal
                        ;
                        ;       La verdadera disposición del disquete es la siguiente :
                        ;
                        ; Pista 0                             Sector
                        ;           1     2     3     4     5     6     7     8     9
                        ; Cabeza +-----+-----+-----+-----+-----+-----+-----+-----+-----+
                        ;   0    |Boot |<======= CCP ========>|<=======  BDOS ========|
                        ;        +-----+-----+-----+-----+-----+-----+-----+-----+-----+
                        ;   1    |====== BDOS ====>|<==============  BIOS  ============>|
                        ;        +-----+-----+-----+-----+-----+-----+-----+-----+-----+
                        ;           10    11    12    13    14    15    16    17    18
                        ;                                 Sector
                        ;
                        ;       Equivalencias para definir tamaño de memoria, dirección base
                        ;       y longitud de los componentes del sistema.
                        ;
0040 =                  Memory$Size     EQU     64      ;Número de Kbytes de RAM
                        ;
                        ;       La longitud del BIOS debe coincidir con la declarada en el BIOS.
                        ;
0900 =                  BIOS$Length     EQU     0900H
                        ;
0800 =                  CCP$Length      EQU     0800H   ;Constante
0E00 =                  BDOS$Length     EQU     0E00H   ;Constante
                        ;
0008 =                  Length$In$K     EQU     ((CCP$Length + BDOS$Length + BIOS$Length) / 1024) + 1
1F00 =                  Length$In$Bytes EQU     CCP$Length + BDOS$Length + BIOS$Length
                        ;
                                IF      NOT Debug
E000 =                  CCP$Entry       EQU     (Memory$Size - Length$In$K) * 1024
                                ENDIF
                                IF      Debug
                        CCP$Entry       EQU     3980H   ;Lectura en la dirección inferior
                                                        ;Esta dirección se escoge de manera que esté antes
                                                        ;  del área en la que inicialmente carga el DDT y 980H
                                                        ;  compone las direcciones de forma similar a los
                                                        ;  valores SYSGEN de manera que la imagen de memoria
                                                        ;  puede ser comprobada con DDT
                                ENDIF

E806 =                  BDOS$Entry      EQU     CCP$Entry + CCP$Length + 6
F600 =                  BIOS$Entry      EQU     CCP$Entry + CCP$Length + BDOS$Length
                        ;
                        ;
                        ;       Características del disco
                        ;
                        ;       Estas equivalencias describen las características físicas
                        ;       del disquete de manera que el programa pueda ir de un sector
                        ;       al siguiente actualizando la pista y poniendo el sector a cero
                        ;       cuando sea necesario.
                        ;
0001 =                  First$Sector$on$Track   EQU     1
0012 =                  Last$Sector$on$Track    EQU     18
0009 =                  Last$Sector$on$Head$0   EQU     9
0200 =                  Sector$Size             EQU     512
                        ;
                        ;
                        ;       Características del controlador
                        ;
```

Figura 7-7. Ejemplo de cargador de arranque en frío del CP/M.

```
                ;       En esta computadora el controlador puede escribir
                ;       varios sectores con una sola orden. Sin embargo,
                ;       para dar un mejor ejemplo, se muestra sólo
                ;       la lectura de un sector cada vez.
                ;
0001 =          Sectors$Per$Read        EQU     1
                ;
                ;
                ;       Cold boot characteristics
                ;
0000 =          Start$Track             EQU     0       ;Valores iniciales de la imagen CP/M
0002 =          Start$Sector            EQU     2       ;= " =
0010 =          Sectors$To$Read         EQU     (Length$In$Bytes + Sector$Size - 1) / Sector$Size
                ;
                ;
                ;

0100                    ORG     100H
                Cold$Boot$Loader:
0100 C34001             JMP     Main$Code       ;Entrada del cuerpo del programa principal
                                                ;Para claridad del programa, las estructuras
                                                ;  de datos se muestran antes
                                                ;  del código ejecutable
000D =          CR      EQU     0DH             ;Retorno de carro
000A =          LF      EQU     0AH             ;Salto de línea
                ;
                Signon$Message:
0103 0D0A43502F         DB      CR,LF,'CP/M Bootstrap Loader'
                        IF      Debug
                        DB      ' (Debug)'
                        ENDIF
011A 0D0A               DB      CR,LF
011C 5665727369         DB      'Version '
0124 3031               DW      Version
0126 20                 DB      ' '
0127 3037               DW      Month
0129 2F                 DB      '/'
012A 3234               DW      Day
012C 2F                 DB      '/'
012D 3832               DW      Year
012F 0D0A00             DB      CR,LF,0
                ;
                ;        Disk Control Tables
                ;
0045 =          Disk$Control$5  EQU     45H     ;Byte de control 5 1/4"
0046 =          Command$Block$5 EQU     46H     ;Apuntador a tabla de control
0043 =          Disk$Status     EQU     43H     ;Terminación del estado
                ;
                ;
                ;       La pista de tablas de órdenes y la DMA$Address pueden utilizarse
                ;       como memoria de trabajo y actualizadas cuando continúa el
                ;       proceso de carga. El sector en la tabla de órdenes no puede
                ;       utilizarse directamente como lo necesita el controlador de
                ;       disco, debe ser el número de sector sobre la cabeza especificada
                ;       (1 -- 9) y más que el número de sector en la pista. Debe usarse,
                ;       por tanto, una variable separada.
                ;
0132 02         Sector:         DB      Start$Sector
                ;
0133 01         Command$Table:  DB      01H     ;Orden -- lectura
0134 00         Unit:           DB      0       ;Unidad número = 0 ó 1
0135 00         Head:           DB      0       ;Número de cabeza 0 ó 1
0136 00         Track:          DB      Start$Track     ;Utilizada como variable de trabajo
0137 00         Sector$on$head: DB      0       ;Convertido a controlador de bajo nivel
0138 0002       Byte$Count:     DW      Sector$Size * Sectors$Per$Read
013A 00E0       DMA$Address:    DW      CCP$Entry
013C 4300       Next$Status:    DW      Disk$Status     ;Apuntador a bloque de estado siguiente
                                                        ;  si las órdenes están encadenadas
013E 4500       Next$Control:   DW      Disk$Control$5  ;Apuntador al siguiente byte de control
                                                        ;  si las órdenes están encadenadas

                Main$Code:
0140 310001             LXI     SP,Cold$Boot$Loader     ;La pila crece hacia abajo debajo del programa
```

Figura 7-7. (Continuación.)

```
0143 210301             LXI     H,Signon$Message        ;Funcionando
0146 CDD901             CALL    Display$Message

0149 213301             LXI     H,Command$Table         ;Apuntar el controlador de disco
014C 224600             SHLD    Command$Block$5         ;  al bloque de comandos

014F 0E10               MVI     C,Sectors$To$Read       ;Fijar contador de sectores
                Load$Loop:
0151 CD7B01             CALL    Cold$Boot$Read          ;Escritura de datos en la memoria
0154 0D                 DCR     C                       ;Decrementar contador

                        IF      NOT Debug
0155 CA00F6             JZ      BIOS$Entry              ;Entrada del BIOS cuando termine la carga
                        ENDIF
                        IF      Debug
                        JZ      0                       ;Arranque caliente
                        ENDIF

0158 213201             LXI     H,Sector                ;Actualización número de sector
015B 3E01               MVI     A,Sectors$Per$Read      ;  añadiendo el número de
015D 86                 ADD     M                       ;  sectores al controlador
015E 77                 MOV     M,A                     ;Guardar resultado
015F 3E13               MVI     A,Last$Sector$On$Track + 1      ;Comprobación de final de pista
0161 BE                 CMP     M
0162 C26E01             JNZ     Not$End$Track

0165 3601               MVI     M,First$Sector$On$Track ;Sí, puesta al comienzo
0167 2A3601             LHLD    Track                   ;Actualización número de pista
016A 23                 INX     H
016B 223601             SHLD    Track

                Not$End$Track:
016E 2A3A01             LHLD    DMA$Address             ;Actualización dirección de DMA
0171 110002             LXI     D,Sector$Size * Sectors$Per$Read
0174 19                 DAD     D
0175 223A01             SHLD    DMA$Address
0178 C35101             JMP     Load$Loop               ;Lectura del bloque siguiente
                ;
                Cold$Boot$Read:                 ;En este punto la descripción de la operación
                                                ;  que se requiere está en las variables
                                                ;  contenidas en la tabla de órdenes a lo
                                                ;  largo de la variable de sector

017B C5                 PUSH    B                       ;Guardar contador de sector en C

                ;------ Cambiar esta rutina para coincidir con el controlador de disco utilizado ------

017C 0600               MVI     B,0                     ;Suposición de cabeza 0
017E 3A3201             LDA     Sector                  ;Obtención de sector requerido
0181 4F                 MOV     C,A                     ;Obtención de copia de él
0182 FE0A               CPI     Last$Sector$on$Head$0+1 ;Comprobación de cabeza 1
0184 DA8B01             JC      Head$0                  ;No
0187 D609               SUI     Last$Sector$on$Head$0   ;Para cabeza 1
0189 4F                 MOV     C,A                     ;Guardar copia
018A 04                 INR     B                       ;Activar cabeza 1
                Head$0:
018B 78                 MOV     A,B                     ;Obtención cabeza
018C 323501             STA     Head
018F 79                 MOV     A,C                     ;Obtención sector
0190 323701             STA     Sector$On$Head

0193 214500             LXI     H,Disk$Control$5        ;Activación del controlador
0196 3680               MVI     M,80H

                Wait$For$Boot$Complete:
0198 7E                 MOV     A,M                     ;Obtención byte de estado
0199 B7                 ORA     A                       ;Comprobación de completo
019A C29801             JNZ     Wait$For$Boot$Complete  ;No
                                                        ;Sí, comprobación de errores
019D 3A4300             LDA     Disk$Status
01A0 FE80               CPI     80H
01A2 DAA701             JC      Cold$Boot$Error         ;Sí, ha ocurrido un error

                ;------ Final de la rutina de lectura ------
```

Figura 7-7. (Continuación.)

```
01A5 C1                 POP     B                       ;Recuperar contador de sectores en C
01A6 C9                 RET
                ;
                Cold$Boot$Error:
01A7 21B001             LXI     H,Cold$Boot$Error$Message
01AA CDD901             CALL    Display$Message         ;Salida de mensaje de error
01AD C34001             JMP     Main$Code               ;Reinicializar cargador
                ;
                Cold$Boot$Error$Message:
01B0 0D0A426F6F         DB      CR,LF,'Bootstrap Loader Error - retrying...',CR,LF,0
                ;
                ;       Equates for Terminal Output
                ;
0001 =          Terminal$Status$Port    EQU     01H
0002 =          Terminal$Data$Port      EQU     02H
                ;
0001 =          Terminal$Output$Ready   EQU     0000$0001B
                ;
                ;
                Display$Message:        ;Visualiza en la consola el mensaje especificado
                                        ;En entrada, HL apunta al flujo de bytes a lanzar
                                        ;a la salida. Termina el mensaje un byte 00H
01D9 7E                 MOV     A,M             ;Obtención byte siguiente del mensaje
01DA B7                 ORA     A               ;Comprobación de terminal
01DB C8                 RZ                      ;Si, retorno a rutina de llamada
01DC 4F                 MOV     C,A             ;Prepare para salida
                Output$Not$Ready:
01DD DB01               IN      Terminal$Status$Port    ;Comprobación de preparado para salida
01DF E601               ANI     Terminal$Output$Ready
01E1 CADD01             JZ      Output$Not$Ready        ;No, esperar
01E4 79                 MOV     A,C                     ;Obtención de carácter de datos
01E5 D302               OUT     Terminal$Data$Port      ;Salida a pantalla

01E7 23                 INX     H               ;Transferir el byte siguiente del mensaje
01E8 C3D901             JMP     Display$Message ;Bucle hasta salida del mensaje completada

                                                ;El cargador basado en PROM comprueba
                                                ;  que los caracteres "CP/M" estén en el sector
                                                ;  de autocarga del disquete antes de transferir
                                                ;  el control a éste
02E0                    ORG     2E0H
02E0 43502F4D           DB      'CP/M'
02E4                    END     Cold$Boot$Loader
```

Figura 7-7. (Continuación.)

como si se tratase de un programa transitorio normal, aunque con menos cambios en la dirección en la que carga la imagen del CP/M del disco. Naturalmente, este mecanismo no es de gran ayuda si el CP/M se introduce por primera vez en un sistema de cálculo. Sin embargo, resulta muy útil si se necesita modificar el cargador o añadir la capacidad de arrancar el sistema desde un nuevo tipo de unidad de disco.

En este caso, el código del cargador deberá cargarse en la posición 0780H, no en la normal 0980H, porque el cargador toma un sector completo de 512 bytes (200H). El mismo principio se aplica para determinar el valor del desplazamiento que se ha de utilizar con la orden "R" del DDT a fin de leer el fichero de autocarga HEX, es decir:

Desplazamiento = dirección de carga − dirección de ejecución

En este caso, los valores son los siguientes:

0680H = 0780H − 0100H

Utilización de MOVCPM para reubicar el CCP y el BDOS

MOVCPM construye una imagen en memoria del CP/M en los lugares correctos para SYSGEN, pero con las instrucciones modificadas para ejecutar en una dirección específica. Dentro de MOVCPM existe no sólo una réplica completa del CP/M, sino también suficiente información para comunicar a MOVCPM qué bytes de qué instrucciones necesitan cambiarse cada vez que la dirección de ejecución de la imagen necesita ser transportada.

MOVCPM, como lo edita Digital Research, contiene cargador y el BIOS para una computadora Intel MDS-800 además del CCP y el BDOS genéricos. A no ser que se posea un MDS-800, todo lo que se utiliza es el CCP y el BDOS. Algunos fabricantes han modificado MOVCPM para que contenga el cargador y el BIOS correctos para sus propias computadoras; para comprobar si se adaptan a una computadora determinada, se aconseja consultar la documentación correspondiente.

Cuando se llama a MOVCPM, se tienen las siguientes opciones:

- MOVCPM<cr>
  MOVCPM volverá a colocar su copia interna del CP/M al principio de la memoria disponible y entonces transferirá el control a esta nueva imagen del CP/M. A menos que el fabricante haya incluido el BIOS correcto en el MOVCPM. Utilizando esta opción se puede originar un estropicio inmediato del sistema.

- MOVCPM nn<cr>
  Similar a la opción anterior, excepto que MOVCPM supone que hay disponibles *nn*K bytes de memoria y volverá a colocar la imagen del CP/M al principio de ella antes de transferir el control. También aquí caerá el sistema a menos que se halle instalado el BIOS correcto en MOVCPM.

- MOVCPM * *<cr>
  MOVCPM ajustará todas las direcciones internas dentro de la imagen de CP/M de forma que esta imagen pueda ejecutarse al principio de la memoria disponible, pero en lugar de poner dicha imagen verdaderamente al principio de la memoria, MOVCPM la dejará en la memoria inferior, en el sitio correcto para que SYSGEN la escriba en un disco. La orden SAVE puede también guardar la imagen en un disco.

- MOVCPM *nn* *<cr>
  MOVCPM procede como en el caso anterior para la opción "* *" excepto que la imagen del CP/M se modifica para ser ejecutada al principio de *nn*K.

MOVCPM tiene un problema fundamental. El valor *nn* indica que el principio de la memoria disponible está calculado, suponiendo que el BIOS es pequeño —menos de 890 (380H) bytes—. Si el BIOS es mayor (como es el caso con el ejemplo de la figura 6-4), entonces habrá que reducir el valor de "*nn*" artificialmente.

La figura 7-8 muestra la relación existente entre el tamaño del BIOS y el valor de "*nn*" para utilizar con el MOVCPM. Muestra también, para diferentes longitudes del BIOS, la dirección base de éste, el valor de desplazamiento que ha de usarse en el DDT para leer en el BIOS en la situación 1F80H (preparatoria para utilizar SYSGEN o PUTCPM para escribirlo), y también las direcciones base para el CCP y el BDOS. La dirección base de este último indica la cantidad de memoria disponible para cargar programas transitorios, porque el CCP puede ser reescrito destructivamente si es necesario.

Los números de la figura 7-8 están basados en la suposición de que se tengan 64K de memoria en la computadora. Si no es este el caso, se procederá de la forma siguiente:

1. Convertir el total de memoria del sistema a hex. Recuérdese que 1K es 1.024 bytes.
2. Determinar la longitud del BIOS en hex.

BIOS Longitud	BIOS Base	DDT Desplazamiento	MOVCPM 'nn'	CCP Base	BDOS Base
600	FA00	2580	64	E400	EC00
A00	F600	2980	63	E000	E800
E00	F200	2D80	62	DC00	E400
1200	EE00	3180	61	D800	E000
1600	EA00	3580	60	D400	DC00
1A00	E600	3980	59	D000	D800
1E00	E200	3D80	58	CC00	D400
2200	DE00	4180	57	C800	D000
2600	DA00	4580	56	C400	CC00
2A00	D600	4980	55	C000	C800
2E00	D200	4D80	54	BC00	C400
3200	CE00	5180	53	B800	C000
3600	CA00	5580	52	B400	BC00
3A00	C600	5980	51	B000	B800
3E00	C200	5D80	50	AC00	B400
4200	BE00	6180	49	A800	B000
4600	BA00	6580	48	A400	AC00
4A00	B600	6980	47	A000	A800
4E00	B200	6D80	46	9C00	A400
5200	AE00	7180	45	9800	A000
5600	AA00	7580	44	9400	9C00
5A00	A600	7980	43	9000	9800
5E00	A200	7D80	42	8C00	9400
6200	9E00	8180	41	8800	9000
6600	9A00	8580	40	8400	8C00
6A00	9600	8980	39	8000	8800

Aparte del valor 'nn' de MOVCPM, todos los demás valores son hexadecimales

Figura 7-8. Direcciones del CP/M para diferentes longitudes de BIOS.

3. Localizar la línea de la figura 7-8, que muestra una longitud de BIOS igual o mayor que la longitud del BIOS que se utilice.
4. Utilizando la orden "H" del DDT, calcular la dirección base del BIOS utilizando la fórmula:

   Memoria del sistema − longitud del BIOS de la figura 7-8
5. Hallar la línea en la figura 7-8 que muestra la misma dirección base del BIOS que el resultado del cálculo anterior. Utilizar esta línea para derivar los otros números relevantes.

Es de gran ayuda utilizar el DDT para examinar la imagen del CP/M en memoria a fin de comprobar que todos los componentes están correctamente situados y, en el caso del CCP y del BDOS, correctamente reubicados.

La figura 7-9 muestra un ejemplo de diálogo de consola en el cual el DDT se utiliza en primer lugar para examinar la imagen en memoria producida por MOVCPM y en segundo lugar para examinar la imagen construida en la utilidad PUTCPMF que se muestra en la figura 7-5.

```
                    Llamada a MOVCPM pidiendo un sistema '63K' y
                    suprimiendo la imagen de la memoria.
A>Movcpm 63 *<cr>
CONSTRUCTING 63k CP/M vers 2.2
READY FOR "SYSGEN" OR
"SAVE 34 CPM63.COM"

                    Almacenar la imagen a partir de la posición 100H. Por
                    convenio el nombre del fichero es CPMnn.COM, de
                    manera que en este caso será CPM63.COM.
A>Save 34 cpm63.com<cr>

                    Llamada a DDT y petición de que lea en CPM63.COM.

A>ddt cpm63.com<cr>
DDT VERS 2.2
NEXT  PC
2300 0100

                    Visualizar la memoria para mostrar los primeros bytes
                    del CCP. Nótense las dos instrucciones JMP (C3H)
                    seguidas por 7FH, 00H, 20H's y la información del
                    Copyright de Digital Research. Esto identifica al
                    programa como el CCP. Nótese que la primera
                    instrucción JMP es a la posición 35CH en el CCP -- de
                    donde se infiere la dirección base del CCP --. En este
                    caso el JMP es a la posición E35C, puesto que esta
                    versión del CCP se ha configurado con base de
                    ejecución en E000H.
-d980,9cf<cr>
0980 C3 5C E3 C3 58 E3 7F 00 20 20 20 20 20 20 20 20 .\..X...
0990 20 20 20 20 20 20 20 20 43 4F 50 59 52 49 47 48         COPYRIGH
09A0 54 20 28 43 29 20 31 39 37 39 2C 20 44 49 47 49 T (C) 1979, DIGI
09B0 54 41 4C 20 52 45 53 45 41 52 43 48 20 20 00 00 TAL RESEARCH  ..
09C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

                    Visualizar los primeros bytes del BDOS. Nótese la
                    instrucción JMP a la posición 1186. Es la instrucción
                    a la que se transfiere el control desde el JMP en la
                    posición 5.
-d1180,118F<cr>
1180 00 16 00 00 09 85 C3 11 E8 99 E8 A5 E8 AB E8 B1 ................
```

Figura 7-9. Utilización del DDT para comprobar imágenes del CP/M.

```
                        Visualizando más arriba en el BDOS se le identifica
                        de forma no ambigua -- existen algunos mensajes de
                        error ASCII.
-d1230,126f<cr>
1230 E8 21 DC E8 CD E5 E8 C3 00 00 42 64 6F 73 20 45 .!........Bdos E
1240 72 72 20 4F 6E 20 20 3A 20 24 42 61 64 20 53 65 rr On  : $Bad Se
1250 63 74 6F 72 24 53 65 6C 65 63 74 24 46 69 6C 65 ctor$Select$File
1260 20 52 2F 4F 24 E5 CD C9 E9 3A 42 EB C6 41 32 C6  R/O$....:B..A2.

                        Visualizar los primeros bytes del BIOS.
                        Nótese el vector de saltos del BIOS -- una serie de
                        instrucciones C3H --. Normalmente la primera instrucción
                        del vector se utiliza para deducir la dirección base
                        del BIOS; en este caso es F600H. Pero no existe una
                        regla fija que diga que el programa de arranque en
                        frío debe estar cerca del vector de saltos del BIOS
                        -- esto es sólo una guía aproximada.
-d1f80<cr>
1F80 C3 B3 F6 C3 C3 F6 C3 61 F7 C3 64 F7 C3 6A F7 C3 .......a..d..j..
1F90 6D F7 C3 72 F7 C3 75 F7 C3 78 F7 C3 7D F7 C3 A7 m..r..u..x..}...
1FA0 F7 C3 AC F7 C3 BB F7 C3 C1 F7 C3 CA F7 C3 70 F7 ..............p.
1FB0 C3 B1 F7 82 F6 00 00 00 00 00 00 6E F8 73 F6 0D ...........n.s..
1FC0 F9 EE F8 82 F6 00 00 00 00 00 00 6E F8 73 F6 3C ...........n.s.<
1FD0 F9 1D F9 82 F6 00 00 00 00 00 00 6E F8 73 F6 6B ...........n.s.k
1FE0 F9 4C F9 82 F6 00 00 00 00 00 00 6E F8 73 F6 9A .L.........n.s..
1FF0 F9 7B F9 1A 00 03 07 00 F2 00 3F 00 C0 00 10 00 .{........?.....
2000 02 00 01 07 0D 13 19 05 0B 11 17 03 09 0F 15 02 ................
2010 08 0E 14 1A 06 0C 12 18 04 0A 10 16 0D 0A 0A 36 ...............6
2020 33 6B 20 43 50 2F 4D 20 76 65 72 73 20 32 2E 32 3k CP/M vers 2.2
2030 0D 0A 00 31 00 01 21 9C F6 CD D3 F7 AF 32 04 00 ...1..!......2..
-^C

                        Como contraste, cargar DDT y realizar petición de
                        carga del programa PUTCPMF5.COM.
A>ddt putcpmf5.com<cr>
DDT VERS 2.2
NEXT  PC
2900 0100

                        Visualizar el cargador especial que comienza en la
                        posición 0780H (compárese con el cargador MDS-800 que
                        está en 0980H). Nótese el mensaje de "funcionando".
-d780,7af<cr>
0780 C3 40 01 0D 0A 43 50 2F 4D 20 42 6F 6F 74 73 74 .@...CP/M Bootst
0790 72 61 70 20 4C 6F 61 64 65 72 0D 0A 56 65 72 73 rap Loader..Vers
07A0 69 6F 6E 20 30 31 20 30 37 2F 32 34 2F 38 32 0D ion 01 07/24/82.

                        Confirmar que el CCP se ha cargado en el lugar
                        correcto. Comprobación de la dirección de la primera
                        instrucción JMP (0E35CH).
-d980,9bf<cr>
0980 C3 5C E3 C3 58 E3 7F 00 20 20 20 20 20 20 20 20 .\..X...
0990 20 20 20 20 20 20 20 20 43 4F 50 59 52 49 47 48         COPYRIGH
09A0 54 20 28 43 29 20 31 39 37 39 2C 20 44 49 47 49 T (C) 1979, DIGI
09B0 54 41 4C 20 52 45 53 45 41 52 43 48 20 20 00 00 TAL RESEARCH  ..

                        Confirmar que el BDOS está también en su lugar.
-d1180,118f<cr>
1180 00 16 00 00 09 85 C3 11 E8 99 E8 A5 E8 AB E8 B1 ................

                        Confirmar que el BIOS se ha cargado en el lugar
                        correcto. Comprobación del primer JMP para tener una
                        idea de la dirección base del BIOS. Nótese el mensaje
                        de "funcionando".
-d1f80<cr>
1F80 C3 F9 F6 C3 0C FE C3 62 F8 C3 78 F8 C3 86 F8 C3 .......b..x.....
1F90 A4 F8 C3 B4 F8 C3 C5 F8 C3 B6 FB C3 0E FB C3 3B ...............;
1FA0 FB C3 41 FB C3 48 FB C3 DE FB C3 F8 FB C3 94 F8 ..A..H..........
1FB0 C3 B0 FB ED 06 00 00 00 42 6E 25 DF 01 B6 DE 02 ........Bn%.....
1FC0 38 00 00 43 50 2F 4D 20 32 2E 32 2E 30 30 20 30 8..CP/M 2.2.00 0
1FD0 37 2F 31 35 2F 38 32 0D 0A 0A 53 69 6D 70 6C 65 7/15/82...Simple
1FE0 20 42 49 4F 53 0D 0A 0A 44 69 73 6B 20 43 6F 6E  BIOS...Disk Con
1FF0 66 69 67 75 72 61 74 69 6F 6E 20 3A 0D 0A 0A 20 figuration :...
2000 20 20 20 20 41 3A 20 30 2E 33 35 20 4D 62 79 74     A: 0.35 Mbyt
2010 65 20 35 22 20 46 6C 6F 70 70 79 0D 0A 20 20 20 e 5" Floppy..
2020 20 20 42 3A 20 30 2E 33 35 20 4D 62 79 74 65 20   B: 0.35 Mbyte
2030 35 22 20 46 6C 6F 70 70 79 0D 0A 0A 20 20 20 20 5" Floppy...
-^C
A>_
```

Figura 7-9. (Continuación.)

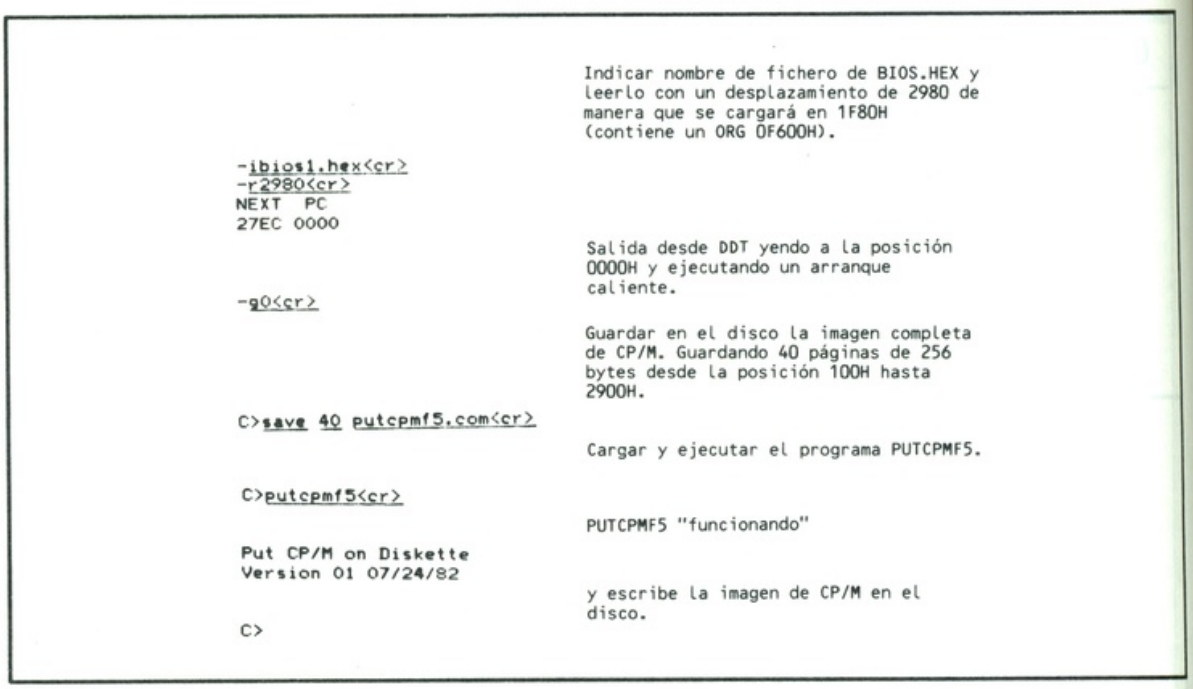
Colocación del conjunto

La figura 7-10 muestra un diálogo de consola, anotado, para la generación completa de un nuevo sistema CP/M. Obsérvese que los nombres de fichero siguiente aparecen en el diálogo:

```
BIOS1.ASM      Figura 6-4.
PUTCPMF5.ASM   Figura 7-5.
BOOTF5.ASM     Figura 7-7.
```

```
                                  Ensamblaje del cargador con el código
                                  fuente y fichero HEX en la unidad C: no
                                  listado de salida.

C>asm bootf5.ccz<cr>
CP/M ASSEMBLER - VER 2.0
02E4
004H USE FACTOR
END OF ASSEMBLY

                                  Ensamblaje del programa PUTCPMF5 (que
                                  escribe CP/M en el disco) con el código
                                  fuente y fichero HEX en la unidad C: no
                                  listado de salida.

C>asm putcpmf5.ccz<cr>
CP/M ASSEMBLER - VER 2.0
01DB
003H USE FACTOR
END OF ASSEMBLY

                                  Ensamblaje del BIOS con el código
                                  fuente y fichero HEX en la unidad C: no
                                  listado de salida.

C>asm bios1.ccz<cr>
CP/M ASSEMBLER - VER 2.0
FE6C
011H USE FACTOR
END OF ASSEMBLY

                                  Montado de la imagen de CP/M. Carga de
                                  DDT y petición de lectura del fichero
                                  almacenado previamente después de
                                  MOVCPM 63*.

C>ddt cpm63.com<cr>
DDT VERS 2.2
NEXT  PC
2300 0100

                                  Indicar nombre de fichero PUTCPMF5.HEX
                                  y leerlo sin ningún desplazamiento (se
                                  cargará en la posición 100H debido al
                                  ORG 100H que contiene). -
                                  iputcpmf5.hex<cr>.

-r<cr>
NEXT  PC
2300 0100

                                  Indicar nombre de fichero BOOTF5.HEX y
                                  leerlo con desplazamiento 680H para que
                                  se cargue a partir de 780H (contiene un
                                  ORG 100H también).

-ibootf5.hex<cr>
-r680<cr>
NEXT  PC
2300 0100
```

Figura 7-10. Diálogo de consola para construcción de sistema.



```
                                    Indicar nombre de fichero de BIOS.HEX y
                                    leerlo con un desplazamiento de 2980 de
                                    manera que se cargará en 1F80H
                                    (contiene un ORG 0F600H).

-ibios1.hex<cr>
-r2980<cr>
NEXT  PC
27EC 0000

                                    Salida desde DDT yendo a la posición
                                    0000H y ejecutando un arranque
                                    caliente.
-g0<cr>

                                    Guardar en el disco la imagen completa
                                    de CP/M. Guardando 40 páginas de 256
                                    bytes desde la posición 100H hasta
                                    2900H.
C>save 40 putcpmf5.com<cr>

                                    Cargar y ejecutar el programa PUTCPMF5.

C>putcpmf5<cr>

                                    PUTCPMF5 "funcionando"

Put CP/M on Diskette
Version 01 07/24/82

                                    y escribe la imagen de CP/M en el
                                    disco.
C>
```

Figura 7-10. (Continuación.)

Mejoras del BIOS
Entrada/salida de caracteres
Estructuras de datos
Entrada/salida de disco
Añadidos normales para el CP/M
Un BIOS mejorado

Escritura de un BIOS mejorado

Este capítulo describe las formas en que puede mejorarse el BIOS para hacer más fácil, más rápido y más versátil el uso del CP/M.

Colóquese un BIOS estándar a trabajar en la computadora y entonces instálense las modificaciones adicionales. Aunque se puede escribir un BIOS mejorado desde el principio, llevará mucho más tiempo conseguir que funcione correctamente.

Al final de este capítulo se incluye una lista completa de las mejoras del BIOS. Es bastante larga: aproximadamente 4.500 líneas de código fuente, con comentarios extensos y nombres de longitud variable para hacerla más clara.

Las secciones siguientes describen los conceptos principales involucrados en la lista de mejoras del BIOS.

Mejoras del BIOS

Las mejoras del BIOS se dividen en dos clases: las que añaden nuevas capacidades y las que extienden características ya existentes.

Algunas mejoras van acompañadas normalmente de programas de utilidad que permiten seleccionar la opción mejorada desde la consola. Por ejemplo, cuando el BIOS se ha mejorado para que contenga un *reloj de tiempo real,* se necesita un programa de utilidad para poner el reloj en hora. Otras mejoras no requieren el soporte de utilidades. Por ejemplo, si los controladores del disco se han mejorado para leer y escribir datos más rápidamente, la mejora es "transparente". Como usuario, uno se beneficia de los resultados de la mejora sin ser consciente de ella.

Enfocado de la forma más simple, el BIOS se enfrenta con dos amplias clases de entrada/salida:

Entrada/salida de caracteres
: Incluye la consola, los auxiliares y los dispositivos de listado.

Entrada/salida de disco
: Esta puede acomodar varios tipos de discos flexibles y rígidos.

Las mejoras de estas áreas no cambian fundamentalmente la forma en que el BDOS y el CCP interaccionan con estos dispositivos. Por el contrario, las mejoras favorecen la forma en que los *controladores de los dispositivos* se enfrentan con éstos. Pueden mejorar la velocidad de manipulación de datos, la forma de tratar los dispositivos externos o el control del usuario sobre el comportamiento del sistema.

El BIOS mejorado del ejemplo posee capacidades que no se encuentran en los sistemas CP/M estándar. Estos pueden agruparse en varias categorías principales:

Entrada/salida de caracteres
: Esta área es probablemente la que más se beneficia de las mejoras. Esto se debe parcialmente a que una amplia cantidad de dispositivos periféricos necesitan ser soportados y en parte a que esta es el área más visible de interacción entre el programador y su computadora. Cualquier mejora que se realice aquí repercutirá inmediata y obviamente en el usuario.

Manejo de errores
: El manejo de errores del CP/M es alarmante por su propia simplicidad. El manejo de errores mejorado proporciona más información acerca de la naturaleza del fallo y entonces proporciona las opciones de reintentar la operación ignorando el error o abortar el programa. Este tópico se trata detalladamente en el capítulo 9.

Sistematización de fecha y hora

Esta es la propiedad de mantener un reloj en hora y la fecha actual. Esto permite a los programas fijar y acceder a fecha y hora. Además, el sistema puede reaccionar al paso del tiempo y se pueden trasladar ciertas operaciones en el terreno del tiempo. Por ejemplo, se pueden fijar límites máximos en el número de segundos o milésimas de segundos que deberá ocupar cada operación y preparar medidas de emergencia si la operación requiere un tiempo demasiado largo.

Asignación de dispositivos lógicos a físicos

La asignación de dispositivos lógicos a físicos del CP/M es muy sencilla. Con las mejoras se puede utilizar cualquier dispositivo de entrada/salida de caracteres como la consola del sistema y enviar datos a varios dispositivos al mismo tiempo.

Entrada/salida de disco

El CP/M conoce sólo el sector de 128 bytes. Incluso con las rutinas de desbloqueo mostradas en la figura 6-4, el rendimiento total del disco puede ser lento. Las prestaciones pueden mejorarse de forma impresionante por medio de la "memoria intermedia de pista" (en la cual se leen pistas enteras y se escriben al mismo tiempo) o utilizando una *memoria de disco* (es decir, utilizando amplias áreas de RAM como si fuera un disco). Esto supone un coste en la medida en que supone aumento de memoria.

Ficheros públicos

El sistema de número de usuario del CP/M necesita progresar para funcionar correctamente en conjunción con grandes discos rígidos.

Preservación de las opciones fijables por el usuario

Un producto secundario de la adición de características al BIOS es que muchas de ellas poseen opciones que pueden alterarse bien desde la consola utilizando un programa de utilidad o desde uno de los programas.

Cada una de estas opciones, una vez fijadas según se prefieran o según los requerimientos del hardware, no cambian normalmente de un día a otro. Por consiguiente, el BIOS deberá ser diseñado de forma que las opciones fijadas por el usuario puedan "congelarse" o preservarse en el disco utilizando el programa de utilidad FREEZE. Todas las variables que recojan estas opciones están reunidas en una sola área y después esta área se escribe en el disco.

Esta área se denomina *bloque de configuración*. En la práctica existen dos bloques de configuración: uno a corto plazo y otro a largo. El bloque a corto plazo no es preservable —se pueden fijar opciones en él, pero no

pueden preservarse después de desconectar la computadora—. La fecha del sistema, por ejemplo, se fija normalmente cada vez que se conecta la computadora y, por consiguiente, se conserva en el bloque a corto plazo. El número de baudio de la impresora, por el contrario, se conserva en el bloque a largo plazo, por lo que puede guardarse de forma permanente.

Un punto de entrada extra del BIOS, la dirección CBGet, se ha construido en el BIOS mejorado de forma que un programa de utilidad pueda situar variables en ambos bloques de configuración. Por ejemplo, cuando una utilidad necesita saber dónde se encuentra la fecha en la memoria, llama a la dirección CBGet utilizando un número de código (específico para la fecha) en un registro. La dirección CBGet devuelve la fecha a la memoria. Si se produce una nueva versión del BIOS con la fecha en un lugar diferente, la dirección CBGet proporcionará la dirección correcta, aunque diferente, devolviéndola al programa de utilidad.

Otras dos variables a las que puede acceder la dirección CBGet pertenecen al propio bloque de configuración. Una es la dirección relativa de inicio del bloque de configuración a largo plazo. La otra es la longitud del bloque a largo plazo. Estas son utilizadas por la utilidad FREEZE cuando se necesita preservar el bloque a largo plazo en un disco. FREEZE deberá: (1) leer en los sectores que contienen el bloque a largo plazo del disco, desde la imagen del BIOS del CP/M en el área reservada del disco, (2) copiar la versión residente en RAM del bloque a largo plazo sobre la versión imagen del disco y, entonces, (3) escribir los sectores en el disco.

La figura 8-1 muestra cómo aparece el bloque a largo plazo en el disco y en la memoria. El tamaño del CCP y del BDOS no cambia aun cuando lo haga el BIOS. Por consiguiente, el sector que contiene el comienzo del BIOS no cambiará. La fórmula (utilizando números decimales) es la siguiente:

```
Sector de comienzo del BIOS+INT(dirección relativa del LTB / 128)
```

da, por tanto, el número de comienzo del sector que se ha de leer. El número de sectores que se han de leer se calcula como sigue:

```
(Longitud del bloque a largo plazo+127) / 128
```

La dirección relativa y la longitud pueden utilizarse para situar el bloque a largo plazo en el BIOS que se ejecuta en RAM.

Entrada/salida de caracteres

Los controladores de entrada/salida de caracteres, mostrados en el BIOS del ejemplo de la figura 8-10, se han mejorado para que posean las siguientes características:

- Un solo juego de subrutinas controla todos los dispositivos de procesamiento de caracteres.
- Preservación de las opciones-estables.
- Redireccionamiento flexible de entrada/salida entre dispositivos lógicos y físicos.
- Controladores de entrada con control de interrupción para obtener la capacidad de que el usuario pueda "mecanografiar en avance".

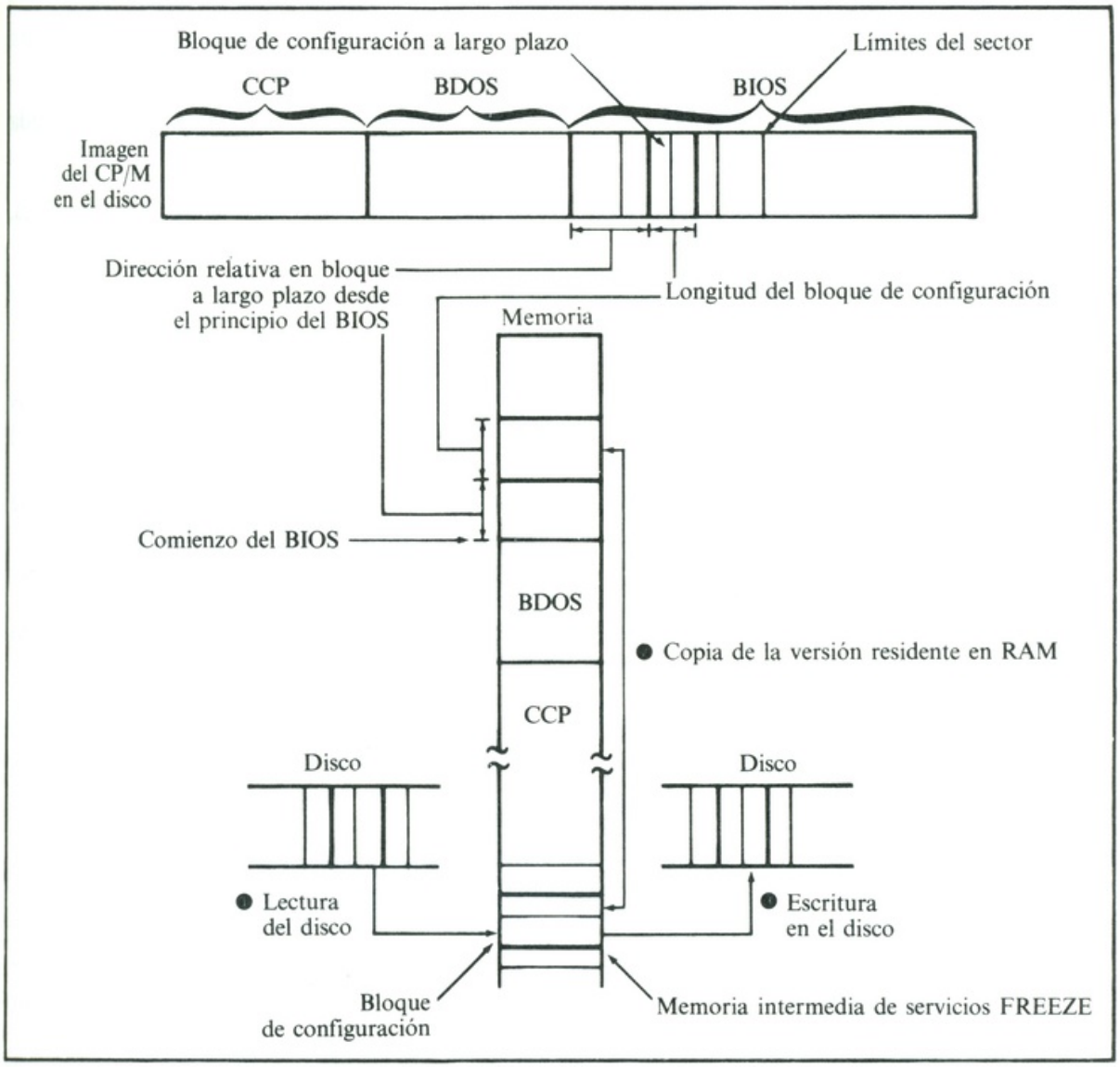


Figura 8-1. Almacenamiento del bloque de configuración a largo plazo.

- Soporte de varios protocolos distintos para evitar la pérdida de datos durante la salida a alta velocidad a las impresoras o otras operaciones.
- Entrada forzada de caracteres en el flujo de entrada de la consola, permitiendo órdenes automáticas.
- Conversión de las teclas de función del terminal en hileras de caracteres útiles.
- Habilidad para reconocer la salida de "secuencias de escape" a la consola y para tomar acciones especiales como resultado de ello.
- Habilidad para leer la hora y fecha actuales como si estuvieran mecanografiadas en la consola.
- "Límite de tiempo" señalando cuando la impresora está ocupada durante demasiado tiempo.

Cada una de estas características se trata en las secciones siguientes, como introducción al ejemplo de programa.

Juego único de subrutinas de control

En los siguientes ejemplos se utiliza un conjunto único de subrutinas para procesar la entrada y salida de todos los dispositivos físicos del sistema.

Esto se ha hecho posible mediante la agrupación de las características de cada dispositivo individual en una tabla denominada *tabla de dispositivos*. Por ejemplo, para obtener un carácter del dispositivo de consola en curso, la dirección de esta tabla de dispositivos se enviará a las subrutinas. Estas, a su vez, utilizarán los valores adecuados de la tabla de dispositivos cuando necesiten acceder a un número de puerto o a un único atributo de dicho dispositivo.

En nuestro ejemplo, los controladores suponen que todos los dispositivos físicos utilizan entrada/salida serie. Para soportar un dispositivo con entrada/salida paralelo, se necesitaría extender la tabla de dispositivos para incluir un campo que hiciera capaces a los controladores de detectar si están operando en un dispositivo serie o paralelo. Probablemente también habría que añadir rutinas diferentes de inicialización de dispositivo y de entrada/salida, más adecuadas a los problemas de tratar con puertos paralelos.

La estructura de la tabla de dispositivos consiste en una serie de instrucciones de equivalencia (EQU). Estas definen el desplazamiento relativo de cada campo de la tabla. Todas las definiciones están expresadas refiriéndose al campo *precedente* de forma que se pueden insertar campos adicionales sin revisar las definiciones para todos los demás campos.

Casos individuales de tablas de dispositivos están definidas como una serie de líneas de definición de bytes (DB) y de definición de palabra (DW). Los controladores reciben la dirección base de la tabla de dispositivos siempre que necesiten hacer algo con un dispositivo. Añadiendo la dirección base a la dirección relativa (definida por la equivalencia), los controladores pueden determinar la dirección real de memoria que contiene el valor requerido. Los contenidos detallados de la tabla de dispositivos se describen más adelante en este capítulo.

Opciones permanentes

Las únicas opciones que necesitan ser preservadas en el bloque de configuración a largo plazo son los valores utilizados para inicializar los chips de hardware. Las demás opciones pueden fijarse durante la ejecución automática del fichero de órdenes cuando el CP/M se carga por primera vez.

Redireccionamiento de entrada/salida entre dispositivos

Como se recordará, el BDOS sólo "está enterado" de los dispositivos *lógicos,* consola, lectora, perforadora y de listado. Utilizando el IOBYTE en la posición 0003H en conjunción con la utilidad STAT, se puede volver a dirigir el BDOS para asignar los dispositivos lógicos a mecanismos físicos específicos. Sin embargo, el redireccionamiento proporcionado por CP/M es más bien primitivo. Permite únicamente cuatro dispositivos físicos por dispositivo lógico. La entrada y la salida de un dispositivo lógico siempre deberán provenir del mismo dispositivo físico. Los datos de salida sólo pueden ser enviados a un único destino, o (utilizando el CONTROL-P) a la consola y al mecanismo de listado.

El sistema de la figura 8-10 soporta hasta 16 dispositivos físicos. Cada uno de ellos puede actuar como consola, lectora, perforadora o dispositivo de listado. La entrada puede venir de cualquier dispositivo individual. La salida puede ser enviada a cualquiera de los dispositivos o a todos ellos. La entrada y salida de cada uno de los dispositivos lógicos están reparadas —es decir, la entrada de consola puede venir del dispositivo físico X, mientras que la salida puede ser enviada a los Y y Z.

El redireccionamiento de los dispositivos puede darse de forma dinámica, tanto desde el interior de un programa como utilizando un programa de utilidad del sistema. Por ejemplo, si se tiene algún dispositivo de entrada especial, el programa puede conectarse momentáneamente para leer entradas de este dispositivo como si fuera la consola, y entonces volver atrás para leer datos de la consola "real".

Este esquema de redireccionamiento se completa definiendo una palabra de 16 bits, denominada *palabra de redirección,* del bloque de configuración a largo plazo para cada uno de los siguientes dispositivos lógicos:

- Entrada de consola.
- Salida de consola.
- Entrada auxiliar (lectora/perforadora).
- Salida auxiliar (lectora/perforadora).
- Entrada de listado (la impresora necesita también enviar datos).
- Salida de listado.

Cada bit de una palabra de redirección dada está asignado a un dispositivo físico. Para la entrada, los controladores utilizan el mecanismo correspondiente al primer bit 1 que encuentran en la palabra de nueva dirección. Para la salida, los controladores envían el carácter para que sea sacado a todos los dispositivos para los que se ha fijado a uno el bit correspondiente.

El programa de ejemplo no selecciona un controlador diferente para cada bit fijado a uno —selecciona una tabla de dispositivos específica y entonces transfiere la dirección base de esta tabla al controlador común utilizado para todas las operaciones de caracteres.

Controladores de entrada con control de interrupción

Con un BIOS de CP/M estándar, los datos carácter se leen desde los chips de hardware sólo cuando el control se transfiere a las subrutinas CONIN o READER. Si estos datos carácter llegan más de prisa de lo que el BIOS puede manejar, ocurre una circulación excesiva de caracteres y se pierden caracteres de entrada.

Usando interrupciones, el hardware puede transferir el control a la rutina de interrupción adecuada siempre que llegue un carácter de entrada. Esta rutina lee el carácter dato y lo sitúa en un área de memoria intermedia para esperar la próxima llamada CONIN o READER, que obtendrá el carácter de esta memoria y lo introducirá en el flujo de datos de entrada.

Los programas de usuarios y el CCP "no piensan" en este proceso, percibiendo sólo que existen caracteres de datos disponibles. Sin embargo, los usuarios serán conscientes del proceso; tendrán la posibilidad de introducir caracteres de datos desde el teclado antes de que el programa esté dispuesto para ello. Esto da a esta técnica otro nombre: "mecanografiado en avance". Aunque esta técnica no altera la velocidad de ejecución de cualquier programa que funcione con CP/M, crea la ilusión de mayor

velocidad, porque las pausas se desvanecen completamente cuando un programa acepta datos. El usuario puede introducir datos a la velocidad que marca el trabajo o el pensamiento sin tener en cuenta la velocidad a la que el programa puede aceptarlos.

El ejemplo contiene el programa necesario para manejar los caracteres que llegan bajo control de interrupciones. Para que sea de aplicación general, el programa supone una estructura de interrupción "plana": es decir, todas las interrupciones de entrada de caracteres originan que el control se transfiera a la misma dirección de la memoria. La dirección está determinada por la arquitectura de interrupción del hardware.

Los esquemas de interrupción más simples utilizan las instrucciones de recomienzo (restart, RST) construidas en el chip CPU 8080. En el esquema RST, el hardware externo interrumpe lo que el chip de CPU está haciendo y fuerza una de las ocho instrucciones del RST dentro del procesador. Cada instrucción RST produce que el procesador ejecute lo que, de hecho, es una instrucción CALL a una dirección previamente determinada de la memoria.

En sistemas más complicados se utilizará un chip de control de interrupción específico (tal como el Intel 8259A). Además, para procurar niveles de prioridad a las interrupciones de manera muy sofisticada (y complicada), el controlador de interrupciones puede transferir el control a direcciones *diferentes,* dependiendo del dispositivo físico que causa la interrupción. Realiza esto forzando a la CPU a que ejecute una instrucción CALL a una dirección diferente para cada dispositivo.

En ambas arquitecturas, es responsabilidad del autor del BIOS inicializar todos los chips del hardware, de forma que la interrupción ocurra en las circunstancias correctas. El programador del BIOS también deberá situar las instrucciones en los sitios correctos de la memoria para recibir el control de una instrucción RST o de la instrucción CALL inventada que emite el controlador de interrupciones.

Algún hardware requiere que la subrutina de servicio de interrupciones le informe tan pronto como la interrupción ha sido atendida y se ha introducido el carácter. El controlador del ejemplo realiza esta tarea.

Esta sección trata el uso de las interrupciones para los controladores de *entrada,* no para los de salida. Hoy en día las microcomputadoras pueden enviar datos mucho más rápidamente que los elementos periféricos pueden manejarlos. Después de algunos minutos de salida, la computadora habrá llenado cualquier memoria intermedia de tamaño razonable —y desde este punto de vista no existe ninguna ventaja en tener un sistema de salida con memoria—. La computadora deberá frenar también la velocidad de los datos hacia los periféricos para cada carácter, aunque ahora esté esperando a poner el carácter en la memoria de salida más que en el periférico.

Una excepción es cuando se tiene una gran cantidad de memoria "sobrante" y una impresora "lenta" (lo que son la mayoría). Un número cada vez mayor de sistemas tienen más de 64K de RAM. El 8080 o el Z80

no pueden direccionar más de esto, pero un sistema de memoria "especial" ("bank switched") puede conmutar bloques de memoria dentro y fuera de este espacio de 64K direcciones.

Usando este truco se puede acceder a memoria "desconocida" para el CP/M, almacenar algunos caracteres en ella, conectar de nuevo a la memoria normal de 64K y devolver el control al llamador de la rutina de salida del BIOS. Cuando el mecanismo físico está dispuesto a aceptar otro carácter de datos de salida del procesador, generará una interrupción. La rutina de servicio de interrupciones accederá entonces a la memoria "secreta", enviando los caracteres al dispositivo y volviendo a conectar a la memoria normal.

Por ejemplo, si se tiene una impresora que imprime a 80 caracteres por segundo y se puede procurar utilizar 64K del banco de memoria conectado, se pueden ahorrar 13 minutos de impresión —o más, si se diseña un esquema para comprimir los espacios en blanco, almacenándolos en la memoria oculta como una secuencia de control especial.

Desde el punto de vista del software, los controladores de entrada con control de interrupción se dividen en dos grandes grupos: la rutina de servicio de interrupciones que lee los caracteres y los acumula en una memoria intermedia, y las rutinas de no interrupción que obtienen los caracteres de ésta y manejan las otras funciones del BIOS tales como el retorno a modo de consola.

La memoria de entrada de caracteres sirve como mecanismo de transferencia entre los dos grupos de subrutinas, aunque la tabla de dispositivos juega también un papel importante.

El programa del ejemplo utiliza una memoria intermedia circular, como se muestra en la figura 8-2.

Los controladores empiezan situando datos al principio de la memoria. Cuando se ha alcanzado el último carácter de ésta, vuelven a situarse al principio y reinician el proceso. Este, por supuesto, considera que los controladores de no interrupción han estado obteniendo datos de la parte anterior de la memoria intermedia, creando espacio para datos de entrada adicionales.

Cada tabla de dispositivos contiene la dirección de la memoria intermedia de entrada, un apuntador de "colocar" (PUT) (para la rutina de servicio de interrupción) y un apuntador de "obtener" (GET) (para la rutina de servicio de no interrupción). Contiene también dos contadores de caracteres: el número total de éstos y el número de caracteres de control en la memoria de entrada. Puede verse cómo los apuntadores de colocar y obtener operan de forma asíncrona. El apuntador de colocar se utiliza cada vez que un carácter que entra genera una interrupción. El apuntador de obtener se utiliza para cada llamada CONIN.

Los apuntadores de obtener y colocar son únicamente valores de un solo byte y se describen con mayor detalle como "desplazamientos relativos". Es

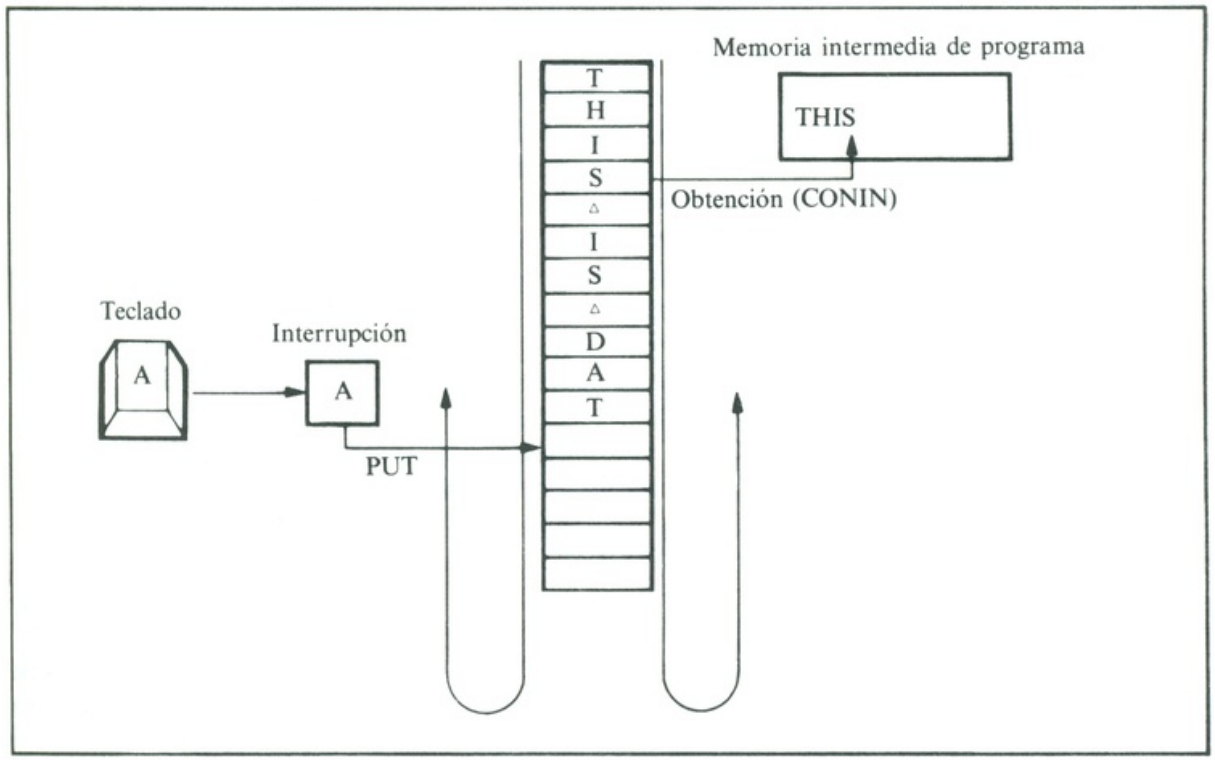


Figura 8-2. Memoria intermedia circular para mecanografiado en avance.

decir, contienen un valor que, cuando se convierte en una palabra y se añade a la dirección base de la memoria intermedia, indicará directamente la posición adecuada dentro de la misma.

Haciendo que la memoria intermedia tenga un número binario de caracteres —32 caracteres, por ejemplo— puede realizarse un truco de programación para que aparezca como si fuera circular. Las tablas de dispositivos contienen un valor enmascarado formado por la longitud del compensador menos uno (longitud−1). Siempre que los apuntadores de obtener y colocar se incrementen en uno (para apuntar a la posición del próximo carácter), se efectúa una operación lógica AND entre el valor actualizado y la máscara (longitud−1). En este ejemplo, si el valor de obtener va desde 31 (la dirección relativa del último carácter de la memoria intermedia) hasta 32 (que sería "fuera del final"), la operación de enmascaramiento volverá a ponerlo a cero (la dirección relativa del primer carácter de la memoria intermedia). Esto evita el tener que comparar los apuntadores para saber cuándo hay que ponerlos a cero.

También es más fácil utilizar la cuenta del número de caracteres de la memoria intermedia, mejor que comparar los apuntadores de obtener y colocar, para distinguir entre una memoria intermedia llena y otra vacía. Para soportar distintos protocolos serie, el controlador debe ser capaz de

reaccionar cuando la memoria intermedia se encuentra cinco caracteres antes de llenarse y cuando está medio vacía. Estas dos condiciones son mucho más fáciles de detectar utilizando una simple cuenta que se incrementa cuando un carácter se coloca en la memoria intermedia y se decrementa cuando un carácter se retira de ella.

La cuenta de caracteres de control se utiliza para tratar con una clase de programas que incesantemente "traga" caracteres, haciendo inútil con ello cualquier continuación de mecanografiado. Un ejemplo es el intérprete BASIC de Microsoft. Cuando está interpretando un programa, se puede introducir un CONTROL-C desde el teclado y el intérprete se parará debidamente. Ello se debe a que se hacen llamadas constantes a CONST (estado de la consola). Si detecta algún carácter de entrada, realiza una llamada a CONIN para introducir el carácter. Un carácter que no sea CONTROL-C se descarta sin más. Por consiguiente, cualquier carácter que se introduzca se "consume" y se destruye el efecto de continuación de mecanografiado.

Para tratar este problema, la rutina CONST mostrada en el ejemplo puede "mentir" acerca del estado de la consola. De esta forma, CONST indicará solamente que los caracteres están esperando en la memoria intermedia de entrada si se recibe un carácter de control. Utiliza la cuenta de caracteres de control para determinar si existen caracteres de control en la memoria intermedia; esta cuenta es incrementada por la rutina de servicio de interrupciones cuando detecta uno, y se decrementa por la rutina CONIN cuando obtiene un carácter de control de la memoria intermedia.

Soporte de protocolos

En este contexto, un protocolo es un esquema para evitar la pérdida de datos que de otra forma podría ocurrir si un dispositivo envía datos más rápidamente de lo que puede aceptar el dispositivo de recepción. Por ejemplo, los protocolos se utilizan para prevenir que la CPU envíe datos a una impresora más rápidamente de lo que ésta puede imprimir los caracteres y mover el papel. Los controladores soportan también los protocolos de entrada, indicando al dispositivo de transmisión cuándo está a punto de llenarse la memoria intermedia de entrada.

Se utilizan dos métodos básicos para implementar protocolos. El primero utiliza las líneas de control que se encuentran en los cables normales de la interfase serie RS-232C. Para datos que salgan de la computadora, se utiliza la señal de terminal de datos (DTR) preparado y para entrada de datos la petición de señal de envío (RTS). Estas señales se adaptan a los estándar eléctricos para la interfase RS-232C; se consideran verdaderos cuando están en algún voltaje positivo entre +3 y +12 voltios, y falsos cuando se encuentran entre −3 y −12 voltios.

El segundo método utiliza caracteres de control ASCII en lugar de señales de control. Dos protocolos separados están soportados por este método. Uno utiliza los caracteres ASCII XON y XOFF. Antes de que el dispositivo de envío (la computadora o algún dispositivo periférico) envíe un carácter de datos, comprueba si se ha recibido un carácter XOFF. Si es así, el emisor esperará un carácter XON. El dispositivo de recepción enviará un XON solamente cuando esté preparado para recibir más datos.

El segundo protocolo utiliza los caracteres ETX (final de transmisión) y ACK (agradecimiento). Este método se usa normalmente sólo cuando se transmiten datos desde la computadora a una impresora con memoria. Está definida la longitud del mensaje (generalmente la mitad del tamaño de la memoria interna de la impresora). Cuando ha salido este número de caracteres, la computadora enviará un carácter ETX. No ocurrirá ninguna otra salida hasta que la computadora reciba un carácter ACK de la impresora.

Los controladores del ejemplo soportan los protocolos alto-para-enviar, DTR, XON/XOFF y ETX/ACK para datos de salida. Para entrada, soporta alto-para-recibir, RTS y el XON/XOFF.

Los protocolos de entrada se invocan cuando a la memoria interna de entrada le faltan cinco caracteres para llenarse. Entonces los controladores envían un carácter XOFF o la señal de bajo voltaje del RTS, o ambas cosas. Solamente cuando la memoria intermedia de entrada se ha vaciado en el 50% de su capacidad, los controladores enviarán un XON o elevan la tensión de la línea de RTS, o ambas cosas.

Como medida de emergencia, si la memoria intermedia de entrada se llena por completo, a pesar de los protocolos, los controladores enviarán un carácter determinado previamente (definido en la tabla de dispositivos) cada vez que descarte un carácter de entrada. Esta señal es normalmente el carácter BELL del ASCII (timbre). Cuando se mecanografía demasiado "en avance", el terminal comienza a zumbar para indicar que los datos son rechazados.

Entrada forzada en el flujo de la consola

Todos los lenguajes de aplicación proporcionan un medio de leer datos desde el teclado de la consola. Esto hace del flujo de entrada a la consola una puerta útil para el sistema. Una simple mejora de las rutinas CONIN/CONST hace fácil "embaucar" al sistema haciendo como si los datos fueran introducidos desde el teclado cuando de hecho vienen de una cadena de caracteres de la memoria.

En el BIOS mejorado tanto CONIN como CONST se han ampliado para comprobar un apuntador en el bloque de configuración a largo plazo, como se muestra en la figura 8-3.

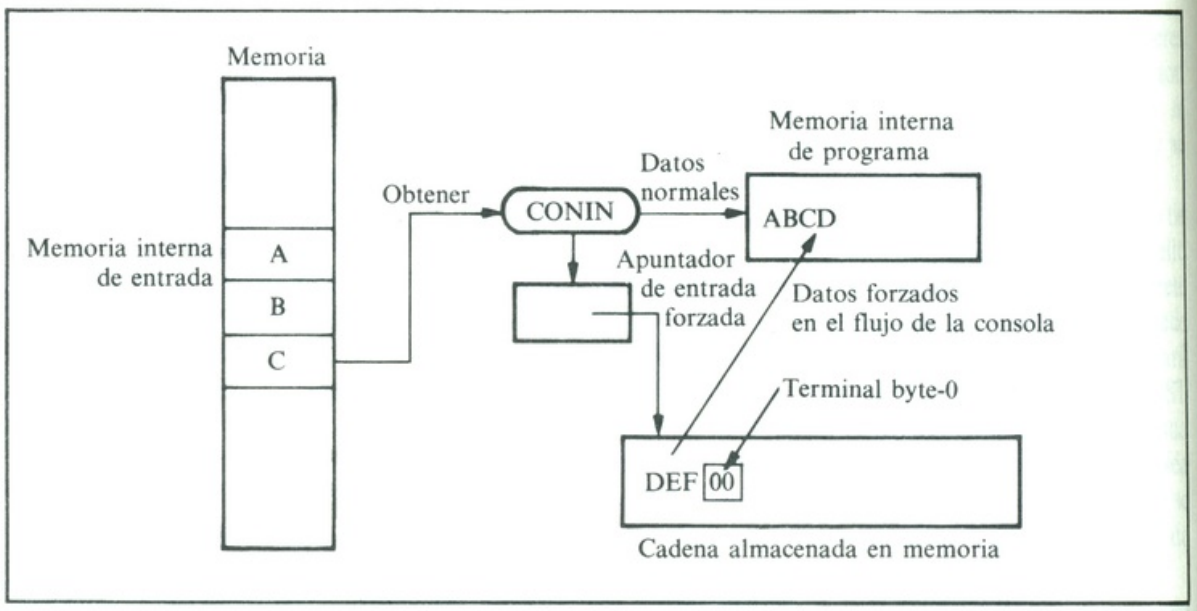


Figura 8-3. CONIN utiliza los datos de la entrada forzada si el apuntador señala un byte distinto de cero.

Si este apuntador señala un byte distinto de cero, entonces este byte se devuelve como si hubiera venido desde el teclado de la consola. El apuntador de entrada forzada se desplaza entonces un byte en la memoria. El proceso de entrada forzada continúa hasta que se encuentra un byte cero.

La entrada forzada sirve para varios propósitos. Puede utilizarse para forzar una o varias órdenes en el sistema cuando éste se pone en marcha por primera vez. En conjunción con un programa de utilidad, puede permitir al usuario que introduzca algunas órdenes de CP/M en una sola línea de mando, inyectando los caracteres según se van realizando las órdenes. También hace posibles las prestaciones que se describen en las dos secciones siguientes.

Soporte de teclas de función en el terminal

Muchos terminales de los que existen actualmente en el mercado tienen teclas de funciones especiales en sus teclados. Cuando se pulsa una de éstas, el terminal emitirá varios caracteres, el primero de los cuales es normalmente el carácter ASCII ESC (escape). Los restantes uno o dos caracteres identifican la tecla de la función específica que se ha pulsado.

Para que estas teclas de función tengan algún uso práctico, un programa de aplicación deberá detectar la secuencia de entrada de escapes y realizar las acciones adecuadas. El problema es que no todos los fabricantes de terminales soportan las secuencias de escape estándar ANSI.

Los controladores del ejemplo evitan este problema proporcionando un método de propósito general, que se muestra en la figura 8-4, de detección de secuencias de escape y de sustitución de una cadena de caracteres definida por el usuario, que se inyecta en la corriente de entrada de la consola como si hubiera entrado del teclado.

Este esquema permite que las teclas función sean utilizadas con mucha flexibilidad, incluso para programas que no han sido diseñados específicamente para aceptar entradas de teclas de funciones.

Sin embargo, existe un bloque de tropiezos. Cuando se recibe un carácter ESCAPE, el programa debe detectar si es el principio de una secuencia de teclas de función o si el usuario está pulsando la tecla de escape

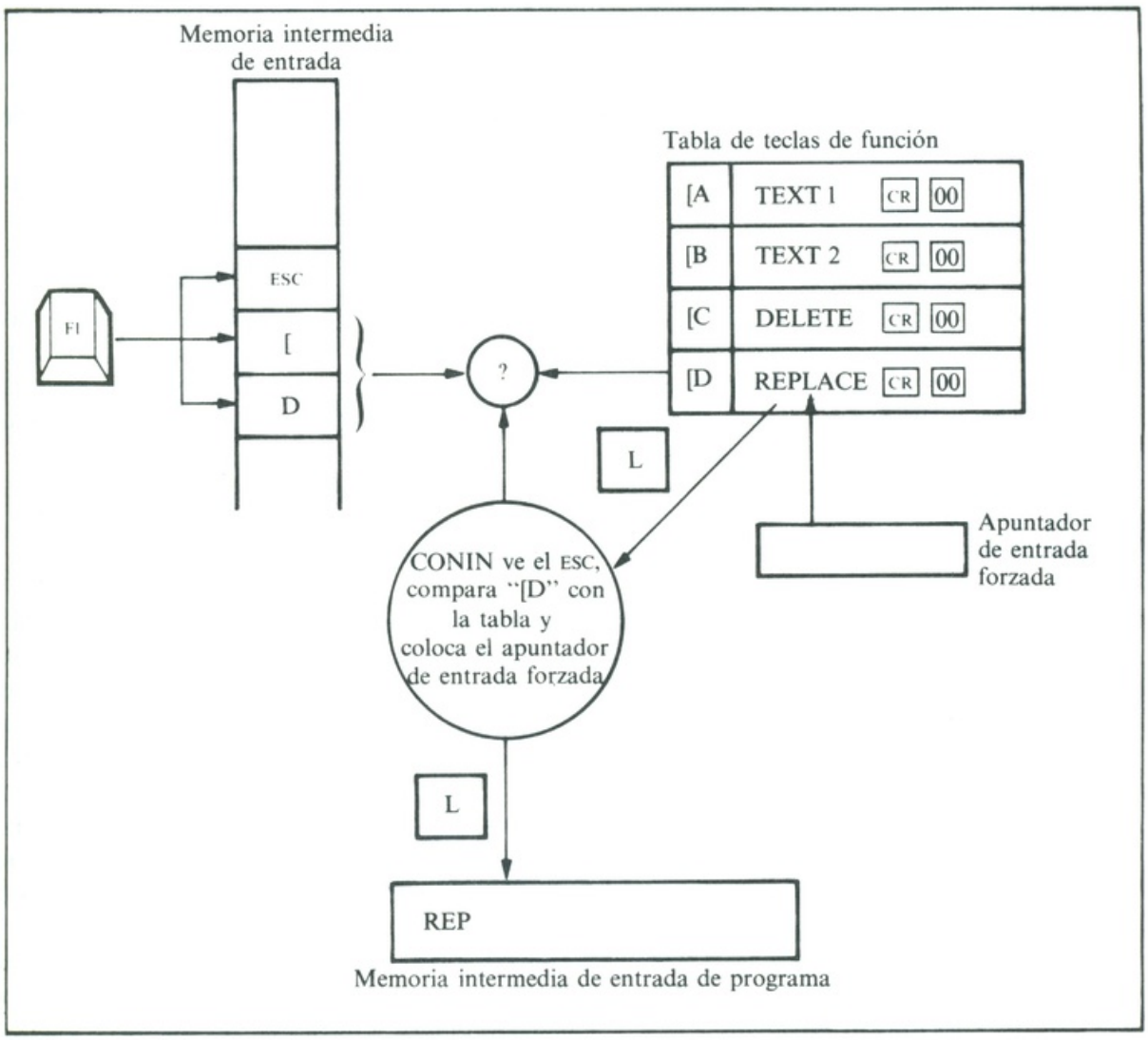


Figura 8-4. CONIN descodifica las teclas de función del terminal.

en el teclado del terminal. En el primer caso, el controlador debe esperar para determinar si una cadena de teclas de función debe ser sustituida por la secuencia de escape. En el último caso, el controlador deberá introducir el carácter de ESCAPE como lo haría con otros caracteres de datos entrantes.

Este reconocimiento sólo puede realizarse trasladándose en el dominio del tiempo. Cuando la rutina CONIN (rutina de no interrupción) obtiene un carácter ESCAPE de una memoria intermedia de entradas, deja pasar aproximadamente 90 milésimas de segundo, tiempo suficiente para que llegue una secuencia de caracteres generada en el terminal. CONIN controla entonces la memoria intermedia de entradas para ver si contiene por lo menos dos caracteres. Si los hay, el controlador los compara en la tabla de teclas de función en el bloque de configuración a largo plazo. Si los caracteres son semejantes a una tecla de función definida, entonces la cadena asociada con ella será inyectada en la corriente de la consola haciendo que el apuntador de entrada forzada le apunte. Si los caracteres no se asemejan a los de la tabla de teclas de función, entonces los caracteres de ESCAPE y subsiguientes se tratarán como caracteres de datos normales.

Si después de 90 milésimas de segundo de retraso no han llegado más caracteres, el carácter ESCAPE se trata como un carácter normal, sobre la base de que ha sido un carácter de escape introducido manualmente más que parte de una secuencia generada por el terminal.

Los controladores del ejemplo muestran el programa necesario y las tablas para las teclas de función que emiten tres caracteres. Se pueden modificar fácilmente para secuencias de dos caracteres, o, si se tiene la suerte de poseer un teclado que utilice los ocho bits de un byte, para reconocer los caracteres de entrada individualmente.

Procesamiento de secuencias de escape de salida

El lado de salida del controlador de consola, la rutina CONOUT, también puede mejorarse para reconocer secuencias de escape. Utiliza una instrucción de JMP vectorizada para mantener la pista al estado actual de los asuntos. El controlador CONOUT obtiene la dirección del vector y le transfiere el control. Normalmente este vector está fijado para el control directo de la rutina de salida de un byte. Sin embargo, si se detecta un carácter de ESCAPE en la corriente de salida, el vector se cambia para transferir el control a una rutina que reconozca el carácter que sigue al ESCAPE. Si el reconocimiento no ocurre, el controlador enviará un ESCAPE seguido del carácter que llegó detrás de él.

Si el segundo carácter es reconocido, entonces el controlador puede transferir el control al procesador correcto de secuencias de escape. Este procesador puede realizar entonces cualquier acción adecuada. También deberá asegurarse de que, cuando se haya acabado todo el proceso, el vector

de salida de consola vuelva de nuevo a procesar los caracteres normales de salida.

Esta técnica se describe de forma más práctica en la sección siguiente, donde se utiliza para fijar y leer la fecha y la hora. Se pueden extender fácilmente tablas de reconocimiento del bloque de configuración a largo plazo para realizar cualquier proceso especial que se necesite, recorriendo desde la alteración de las palabras de redireccionamiento de entrada/salida hasta cambiar cualquier otra variable del sistema o programando hardware especial en la computadora.

Hay que tener cuidado de no incluir ningún valor binario puro en la secuencia de caracteres que sale de la rutina CONOUT. Si se intenta enviar un valor 09H (carácter TAB) por medio del BDOS, éste lo expandirá gratuitamente en varios espacios. Si se necesita enviar cierta configuración de bits, tal como la palabra de redireccionamiento de entrada/salida, se dividirá en una serie de valores de 7 bits de longitud. Entonces se envía en un byte que tenga el bit más significativo a 1. Un valor 09H se convertirá entonces en 89H, previniéndose así la expansión del BDOS a espacios vacíos.

Lectura de fecha y hora desde la consola

Por el momento, dejaremos de lado la cuestión de cómo se introducen en el sistema la fecha y la hora. Como la fecha y la hora están almacenadas en el bloque de configuración a corto plazo (ya que no hay necesidad de salvarlos de una sesión de trabajo a la siguiente), todo lo que necesita el BIOS es ser capaz de reconocer la petición de un programa de aplicaciones que haya de leer la fecha o la hora y entonces fijar el apuntador de entrada forzada a la cadena adecuada de la memoria. Ambas cadenas de fecha y hora están acabadas por un LINE FEED seguido de un byte 00.

Esta secuencia de acontecimientos se muestra en la figura 8-5.

Como puede verse, la salida de caracteres "ESC d" a CONOUT hace que señale el apuntador de entrada forzada en la fecha de la memoria. Las llamadas subsiguientes a CONIN aportan los caracteres de la fecha en el programa como si hubieran sido introducidos en el teclado.

Centinela límite de tiempo en impresora

No existe previsión en el CP/M para tratar con un dispositivo de hardware que por una u otra razón no esté disponible permanentemente. A menos que se tomen medidas especiales en los controladores, el sistema se detendrá en un bucle, leyendo el estado y comprobando que los dispositivos periféricos estén preparados.

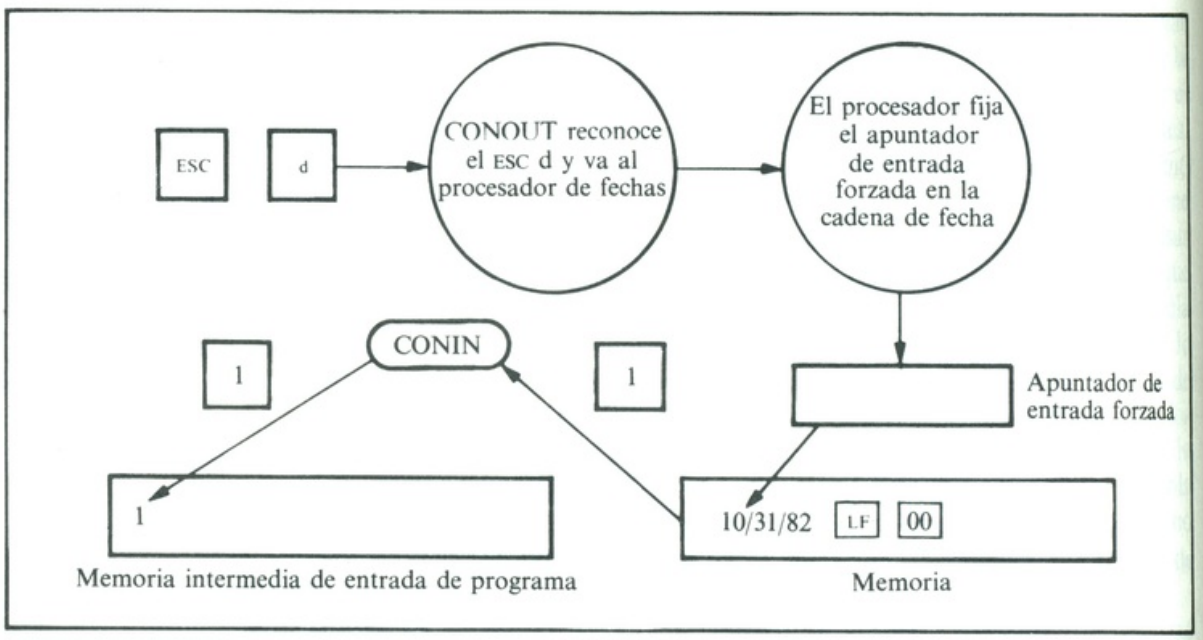


Figura 8-5. Secuencias de escape enviadas a CONOUT permiten que CONIN lea la fecha.

El programa mejorado del ejemplo muestra un esquema utilizando un reloj de tiempo real, que puede detectar cuándo un dispositivo tal como una impresora deja de estar disponible durante más de 30 segundos. Al detectar esta situación, el programa envía un mensaje a todos los dispositivos de la consola que no están siendo utilizados también como impresora. Este tipo de salida se necesita para evitar "lazos destructivos" cuando una impresora que no está disponible genere un mensaje que no pueda enviarse por no estar dispuesta la impresora.

El programa que realiza la función de tiempo se conoce con el nombre de *cronometrador centinela*. Cada vez que el reloj de tiempo real hace "tic-tac", la rutina de servicio de interrupción controla el valor del centinela. Si este valor es distinto de cero, se decrementa. Si el cronometrador de centinela alcanza el cero, superando el tiempo permitido, los controladores compondrán un mensaje en la consola indicando que la impresora ha estado ocupada durante demasiado tiempo. El usuario tiene entonces la opción de preparar la impresora e intentar de nuevo enviar datos, ignorando el error y arrastrándolo, o abortar el programa realizando una puesta a cero del sistema con el BDOS (función 0).

Aunque el envío de un mensaje de error a la consola parece sencillo, resulta complicado si la salida de la consola se dirige hacia la propia impresora ofensora. Los controladores intentan resolver este problema enviando el mensaje sólo a aquellos dispositivos que se utilizan como consolas y *no* como impresoras. Si todas las consolas se utilizan como

dispositivos de impresión, el controlador enviará el mensaje al dispositivo 0 —normalmente la consola principal.

Conservación de hora y fecha

El CP/M no está previsto para conservar la hora y la fecha actuales en el sistema. El ejemplo mejorado muestra cómo mantener la hora y el día y la fecha actual en el bloque de configuración a corto plazo utilizando secuencias de escape enviadas a la consola: (1) para fijarlo en los valores correctos, y (2) para "leerlos" desde el flujo de entrada de la consola.

El ejemplo presupone que el sistema tiene un chip de hardware que puede ser programado para generar una interrupción cada 1/60 de segundo (16.666 milésimas de segundo), lo que significa tener un contador de divisiones para medir los segundos transcurridos. Desde luego, si la computadora tiene un reloj de tiempo *real* que se pueda leer y del cual se puedan obtener las horas, minutos y segundos actuales, el programa será muy sencillo. Sin embargo, se necesitará tener el reloj generando interrupciones periódicas, con el fin de utilizar el dispositivo centinela para cronometrar las operaciones de la impresora y del disco.

El tiempo se conserva en caracteres ASCII, utilizando otra tabla de control ASCII para determinar cuándo hay "acarreo y puesta a cero". Cambiando dos bytes de esta tabla, se mantendrá el tiempo en formato de 12 ó 24 horas.

La fecha se almacena simplemente en una cadena. El programa del ejemplo no intenta asegurar que la fecha sea válida, ni actualizarla después de la medianoche. Esto podría hacerse fácilmente por medio del BIOS, pero consumiría una gran cantidad de programa.

Cronometrador centinela

Al tener una fuente periódica de interrupciones, se abre también la puerta a la construcción de un cronometrador centinela para emergencias. Esto consiste nada más que en un contador de 16 bits. Cada vez que se interrumpe el reloj de tiempo real, o hace tic-tac, la rutina de servicio de interrupciones controla el valor del centinela. Si ya está a cero, no ocurre nada más —el centinela no está en uso—. Si no es cero, la rutina decrementa el valor en uno. Si resulta un valor cero, la rutina de servicio de interrupciones llama a una dirección previamente determinada. Esta será la dirección de alguna rutina de servicio de interrupciones de emergencia que puede entonces tomar una acción especial, tal como investigar la causa del tiempo excedido.

La rutina de centinela posee una subrutina de nivel de no interrupción

asociada a ella. Llamando a esta subrutina de fijación de centinela se consigue un medio de fijar el contador a un número determinado previamente de "tic-tacs" de reloj de tiempo real y la dirección a la que se transferiría el control si la cuenta alcanza el cero.

Habiendo llamado la subrutina de fijación del centinela, el controlador podrá situarse en modo bucle, con posibilidad de interrupciones, esperando que ocurra algún acontecimiento. Si éste ocurre antes de que la cuenta del centinela sea cero, el controlador deberá llamar a la rutina de fijación de centinela de nuevo para fijar la cuenta atrás hasta cero, y con ello desconectar el centinela.

El cronometrador de centinela puede utilizarse para detectar impresoras que están ocupadas durante demasiado tiempo o controladores de disco que tardan demasiado en completar una acción, bien porque tienen un fallo de hardware o porque el usuario no ha cargado el disco en la unidad.

Estructuras de datos

Como ya se ha establecido, cada dispositivo de entrada/salida de caracteres tiene su propia tabla de dispositivo que describe todas sus características particulares.

La otra estructura mayor de datos son los bloques de configuración —a corto y a largo plazo.

Esta sección describe cada campo de estas estructuras de datos.

Tabla de dispositivos

La figura 8-6 muestra los contenidos de una tabla de dispositivos. Más adecuadamente, muestra una serie de equivalencias que definen los desplazamientos de cada campo en la tabla de dispositivos. Los controladores reciben la dirección base de una tabla de dispositivos específica. Acceden entonces a cada campo mediante la adición del desplazamiento requerido a esta dirección base.

La primera parte de la tabla de dispositivos está dedicada al aspecto físico del dispositivo, definiendo qué números de puerto han de ser utilizados para comunicarse con él. Los controladores necesitan saber los distintos números de puerto porque cada uno se utiliza para una función particular. Dependiendo del hardware, cada número de puerto podría ser diferente; sin embargo, con chips estándar Intel o Zilog, se encuentra frecuentemente que el mismo número de puerto se utiliza para varias funciones. Los controladores necesitan también saber qué configuraciones de bits son de esperar cuando se leen algunos puertos y qué valores enviar a los mismos para obtener resultados particulares.

```
                ;       Los controladores utilizan una tabla de dispositivo para cada uno
                ;       de los dispositivos físicos a los que sirven. Las equivalencias
                ;       que siguen se utilizan para acceso a los distintos campos de la tabla
                ;       de dispositivo.
                ;
                ;                               Número de puerto y bits de estado
0000 =          DT$Status$Port          EQU     0               ;Número puerto estado dispositivo
0001 =          DT$Data$Port            EQU     DT$Status$Port+1
                                                ;Número de puerto dispositivo de datos
0002 =          DT$Output$Ready         EQU     DT$DataPort+1
                                                ;Máscara de estado salida disponible
0003 =          DT$Input$Ready          EQU     DT$Output$Ready+1
                                                ;Máscara de estado entrada disponible
0004 =          DT$DTR$Ready            EQU     DT$Input$Ready+1
                                                ;DIR disponible para envío de máscara
0005 =          DT$Reset$Int$Port       EQU     DT$DTR$Ready+1
                                                ;Número de puerto utilizado para puesta a cero
                                                ;  de una interrupción
0006 =          DT$Reset$Int$Value      EQU     DT$Reset$Int$Port+1
                                                ;Valor lanzado para puesta a cero de interrupción
0007 =          DT$Detect$Error$Port    EQU     DT$Reset$Int$Value+1
                                                ;Número de puerto para detección de error
0008 =          DT$Detect$Error$Value   EQU     DT$Detect$Error$Port+1
                                                ;Máscara para detección de errores (paridad, etc.)
0009 =          DT$Reset$Error$Port     EQU     DT$Detect$Error$Value+1
                                                ;Salida al puerto para desactivar error
000A =          DT$Reset$Error$Value    EQU     DT$Reset$Error$Port+1
                                                ;Valor lanzado para desactivar error
000B =          DT$RTS$Control$Port     EQU     DT$Reset$Error$Value+1
                                                ;Puerto de control para disminuir RTS
000C =          DT$Drop$RTS$Value       EQU     DT$RTS$Control$Port+1
                                                ;Valor, en salida, para bajar RTS
000D =          DT$Raise$RTS$Value      EQU     DT$Drop$RTS$Value+1
                                                ;Valor, en salida, para aumentar RTS
                ;
                ;                       Estado lógico del dispositivo (incluido protocolos)
000E =          DT$Status               EQU     DT$Raise$RTS$Value+1
                                                ;Bits de estado
0001 =          DT$Output$Suspend       EQU     0000$0001B      ;Salida suspendida pendiente de la acción
                                                ;  del protocolo
0002 =          DT$Input$Suspend        EQU     0000$0010B      ;Entrada suspendida hasta que se vacíe
                                                ;  la memoria intermedia
0004 =          DT$Output$DTR           EQU     0000$0100B      ;La salida utiliza DTR alto-para-enviar
0008 =          DT$Output$Xon           EQU     0000$1000B      ;La salida utiliza Xon/Xoff
0010 =          DT$Output$Etx           EQU     0001$0000B      ;La salida utiliza Etx/Ack
0020 =          DT$Output$Timeout       EQU     0010$0000B      ;La salida utiliza límite de tiempo
0040 =          DT$Input$RTS            EQU     0100$0000B      ;La entrada utiliza RTS alto-para-recibir
0080 =          DT$Input$Xon            EQU     1000$0000B      ;La entrada utiliza Xon/Xoff
                ;
000F =          DT$Status$2             EQU     DT$Status+1     ;Byte de estado secundario
0001 =          DT$Fake$Typeahead       EQU     0000$0001B      ;Petición a Input$Status para que devuelva
                                                                ;  "Dato preparado" cuando los
                                                                ;  caracteres de control están en la
                                                                ;  memoria intermedia de entrada
                ;
0010 =          DT$Etx$Count            EQU     DT$Status$2+1
                                                ;Número de caracteres enviados en protocolo ETX
0012 =          DT$Etx$Message$Length   EQU     DT$Etx$Count+2
                                                ;Longitud del mensaje especificado
                ;
                ;                       Valores de la memoria intermedia de entrada
0014 =          DT$Buffer$Base          EQU     DT$Etx$Message$Length+2
                                                ;Dirección de la memoria intermedia de entrada
0016 =          DT$Put$Offset           EQU     DT$Buffer$Base+2
                                                ;Desplaz. para puesta de caracteres en la mem. int.
0017 =          DT$Get$Offset           EQU     DT$Put$Offset+1
                                                ;Desplaz. para obtención de carac. de la mem. int.
0018 =          DT$Buffer$Length$Mask   EQU     DT$Get$Offset+1
                                                ;Longitud de la memoria intermedia - 1
                                                ;Nota: La longitud de la memoria intermedia debe ser
                                                ;  siempre un número binario: 32, 64 ó 128
                                                ;Esta máscara se vuelve entonces:
                                                ;   32 ->  31 (0001$1111B)
                                                ;   64 ->  63 (0011$1111B)
                                                ;  128 -> 127 (0111$1111B)
```

Figura 8-6. Equivalencias de tabla de dispositivos.

```
                                        ;Después de que el desplazamiento de la
                                        ; obtención/puesta se ha incrementado es operado
                                        ; (AND) con la máscara para ponerlo a cero cuando
                                        ; se alcanza el final de la mem. int.
0019 =      DT$Character$Count    EQU   DT$Buffer$Length$Mask+1
                                        ;Contador del número de caracteres en curso
                                        ;  en la memoria intermedia
001A =      DT$Stop$Input$Count   EQU   DT$Character$Count+1
                                        ;Parada de la entrada cuando el contador
                                        ;  alcanza este valor
001B =      DT$Resume$Input$Count EQU   DT$Stop$Input$Count+1
                                        ;Reanudar la entrada cuando el contador
                                        ;  alcanza este valor
001C =      DT$Control$Count      EQU   DT$Resume$Input$Count+1
                                        ;Cuenta del número de caracteres de control
                                        ;  en la memoria intermedia
001D =      DT$Function$Delay     EQU   DT$Control$Count+1
                                        ;Número de pulsos de reloj de retraso para llegada
                                        ;  de todos los caracteres después de pulsar
                                        ;  la tecla de función
001E =      DT$Initialize$Stream  EQU   DT$Function$Delay+1
                                        ;Dirección del flujo de datos necesario para
                                        ;  inicializar este dispositivo
```

Figura 8-6. (Continuación.)

La distribución de la tabla de dispositivos y la forma en que se declaran las equivalencias están diseñados para hacer fácil el cambio de contenidos de la tabla para atender cualquier requerimiento especial. Los campos de la primera sección de la tabla de dispositivos se tratan en la sección siguiente.

DT$Status$Port El controlador lee este puerto para determinar si el chip de hardware tiene datos de entrada disponibles para ser introducidos en la computadora o si el chip es capaz de aceptar otros caracteres de datos para salida al dispositivo físico.

DT$Data$Port El controlador lee desde este puerto para acceder al próximo carácter de datos desde el dispositivo físico. El controlador escribe también en este puerto para enviar el próximo carácter de datos al dispositivo.

Si el hardware de la computadora requiere que el puerto de entrada de datos tenga un número diferente del puerto de salida de datos, habrá que alterar la programación en las equivalencias de la tabla de dispositivos, así como hacer los cambios necesarios en las subrutinas de entrada y salida del cuerpo del programa.

DT$Output$Ready Esta es la máscara de bits que con el controlador hará la operación AND con el estado del dispositivo en curso (obtenido leyendo el DT$Status$Port) para ver si el dispositivo está preparado para aceptar otro carácter de salida. Supone que el dispositivo está dispuesto si el resultado de la instrucción AND es distinto de cero. Puede ocurrir que haya que cambiar alguna instrucción JNZ (salto si es distinto de cero) por la instrucción JZ (salto si es cero) si el dispositivo hardware utiliza lógica invertida, con bits en el byte de estado puestos a 0 para indicar que el dispositivo puede aceptar otro carácter para salida.

Obsérvese que la comprobación del estado se refiere sólo al chip de salida —completamente separado de la cuestión de si los elementos periféricos están a su vez dispuestos a aceptar datos.

DT$Input$Ready Esta es la máscara de bits con la que el controlador realizará la operación AND con el estado del dispositivo en curso para ver si existe algún carácter de datos de entrada. Los controladores suponen de nuevo que si el resultado del AND es distinto de cero, un carácter de datos de entrada está esperando para ser leído desde el puerto de datos. Se necesitará realizar cambios similares a los de las rutinas de salida descritas en la sección anterior si el hardware utiliza lógica invertida (bit a 0 significa datos de entrada).

DTDTRReady DTR representa el *terminal de datos preparado*. Se refiere a una de las líneas de control conectadas desde el dispositivo periférico al chip de entrada/salida (por medio de otros varios circuitos integrados). Los controladores, como opción, sólo enviarán datos al dispositivo cuando la señal DTR esté a tensión positiva. Si el periférico, para parar el flujo de caracteres de datos que se le envían, disminuye la señal DTR a una tensión negativa, los controladores esperarán. Una vez el DTR se ha hecho de nuevo positivo, los controladores reanudarán el envío de datos. Muchos dispositivos de copia dura utilizan este esquema para permitirse a sí mismos imprimir los datos recibidos de la computadora. Tendrán que poner el DTR a su valor bajo durante algunos segundos; por ejemplo, mientras realizan el movimiento del papel.

El valor en este campo es una máscara de bits que los controladores utilizan en el estado de dispositivo para determinar el estado de la señal de control de terminal de datos preparado.

DT$Reset$Int$Port Como el lado de entrada de los controladores utiliza interrupciones, cuando un carácter de entrada está preparado para ser introducido por la CPU, el hardware genera una señal de interrupción y el control se transfiere a la rutina del servicio de interrupción. Esta rutina "atiende" la interrupción leyendo el carácter de datos de entrada, salvándolo en la memoria, y transfiriendo entonces el control de nuevo a lo que estaba siendo ejecutado cuando ocurrió la interrupción.

Los chips controladores de interrupciones más complicados (tales como el Intel 8259A) deben ser avisados apenas se ha atendido una interrupción dada, de forma que puedan permitir atender a cualquier interrupción que tenga prioridad más baja que pueda estar esperando.

Este campo contiene el número de puerto que será utilizado para "poner a cero" la interrupción, o más correctamente, para indicar el final de atención a la interrupción previa.

DT$Reset$Int$Value Este es el valor que será enviado al DT$Reset$Int$Port para decir al hardware que la atención a la interrupción previa ha sido completada.

DT$Detect$Error$Port Antes de que el controlador intente leer cualquier dato de entrada del DT$Data$Port, comprueba si ha ocurrido cualquier error de hardware. Lo efectúa leyendo el estado de este puerto.

DT$Detect$Error$Value El byte de estado que se introduce desde el DT$Detect$-Error$Port realiza el AND con este valor. Si el resultado es distinto a cero, el controlador considera que ha ocurrido un error.

DT$Reset$Error$Port Si ha ocurrido un error, el controlador envía un valor de puesta a cero del error a este número de puerto.

DT$Reset$Error$Value Este es el valor que será enviado al DT$Reset$Error$Port para poner a cero un error.

DTRTSControl$Port Los controladores utilizan este número de puerto para controlar la línea de petición de envío si se selecciona la opción de protocolo RTS.

DT$Drop$RTS$Value Este valor se envía al puerto de control del RTS para disminuir la tensión en la línea del RTS de forma que los dispositivos externos dejen de enviar datos a la computadora.

DT$Raise$RTS$Value Este valor se envía para crear la línea de RTS de forma que el dispositivo externo reanude el envío de datos a la computadora.

DT$Status Este es el primero de los dos bytes de estado. Contiene bits que están puestos a 1 para indicar las siguientes condiciones:

DT$Output$Suspend
Como figura en el protocolo, el dispositivo suspende la recepción de cualquier carácter de salida.

DT$Input$Suspend
Como figura en el protocolo, el dispositivo ha sido solicitado para que no envíe más caracteres de entrada.

DT$Output$DTR
El controlador mantendrá el protocolo DTR alto-para-enviar para los datos de salida.

DT$Output$Xon
El controlador mantendrá el protocolo XON/XOFF para datos de salida.

DT$Output$Etx
El controlador mantendrá el protocolo ETX/ACK para datos de salida.

DT$Input$RTS
El controlador mantendrá el protocolo RTS alto-para-recibir para datos de entrada.

DT$Input$Xon
El controlador mantendrá el protocolo XON/XOFF para datos de entrada.

DT$Status$2 Este es otro byte de estado, con el siguiente significado de sus bits:

DT$Fake$Typeahead
CONST "mentirá" acerca de la disponibilidad de los caracteres de entrada en la consola. Indicará solamente que hay datos esperando si son caracteres de control distintos a CARRIAGE RETURN, LINE FEED o TAB en la memoria de entrada.

DTEtxCount Este valor se utiliza sólo por el protocolo ETX/ACK. Es el valor del número de caracteres enviados en el mensaje. Cuando este total alcanza la longitud del mensaje definida, entonces el controlador enviará un carácter ETX y suspenderá cualquier otra salida.

DTEtxMessage$Length Este valor es la longitud de mensaje definida para el protocolo ETX/ACK. Se utiliza para poner a cero el DTEtxCount.

DT$Buffer$Base Esta es la dirección del primer byte de la memoria intermedia del dispositivo de entrada.

DTPutOffset Este byte contiene el desplazamiento relativo indicando cuándo se "pone" el siguiente carácter de entrada en la memoria intermedia. El byte debe convertirse en un valor y sumado a la dirección DTS$Buffer$Base para obtener la posición absoluta en memoria.

DTGetOffset Este byte contiene el desplazamiento relativo indicando cuándo se "obtendrá" el próximo carácter en la memoria intermedia de entrada.

DT$Buffer$Length$Mask Este byte contiene la longitud de la memoria intermedia menos uno. La longitud de ésta deberá ser siempre un número binario (8, 16, 32, 64...). Por consiguiente, uno menos que la longitud forma un valor máscara. Tanto el desplazamiento para obtener como para colocar, después de ser incrementados, están enmascarados con este valor. Cuando el

desplazamiento alcanza el final de la memoria intermedia, esta operación de enmascaramiento lo pone a cero automáticamente.

DT$Character$Count Este es el valor del número total de caracteres en la memoria intermedia. Se incrementa por la rutina del servicio de interrupciones cada vez que se sitúa un carácter en la memoria intermedia, y se decrementa por la rutina CONIN cada vez que obtiene un carácter.

CONST utiliza este valor para determinar si todos los caracteres están disponibles para entrada.

DT$Stop$Input$Count Cuando las rutinas del servicio de interrupción detectan que el DT$Character$Count es igual a este valor (normalmente la longitud de la memoria intermedia menos cinco), los controladores invocarán al protocolo de entrada seleccionado, rebajando el RTS o enviando XOFF, para cerrar el flujo de datos de entrada.

DT$Resume$Input$Count Cuando la rutina CONIN detecta que DT$Character-$Count se ha hecho igual a este valor, los controladores invocarán de nuevo al protocolo de entrada seleccionado, creando un RTS o enviando XON, para reanudar la recepción de datos de entrada.

DT$Control$Count Este es el valor del número de caracteres de control en la memoria intermedia de entrada. Los caracteres CARRIAGE RETURN, LINE FEED y TAB no están incluidos en este valor. Se incrementa por la rutina del servicio de interrupción y se decrementa por CONIN. CONST utiliza el valor cuando el modo DT$Fake$Typeahead está activo; indicará solamente que los caracteres están esperando en la memoria de entrada si el valor de control es distinto a cero.

DT$Function$Delay Este es el número de tic-tacs de reloj que estaría permitido que transcurriera después de que se ha detectado el primer carácter de una secuencia de escape entrante. Permite que haya tiempo para que lleguen los restantes caracteres de la secuencia de escape, suponiendo que éstos son emitidos por un terminal a la máxima velocidad. Normalmente, ésta corresponde a un retraso de aproximadamente 90 milésimas de segundo.

DT$Initialize$Stream Esta es la dirección del primer byte de una cadena. Esta cadena tiene el siguiente formato:

```
DB  ppH          Número de puerto
DB  nnH          Número de bytes que se han de enviar
DB  vvH,vvH...   Bytes de inicialización a enviar al número de puerto especificado
```

Esta secuencia puede repetirse tantas veces como sea necesario, con un número de "puerto" 00H actuando como terminal.

Entrada/salida de disco

Los controladores del ejemplo muestran las tres principales mejoras de entrada/salida del disco:

- Memorización de pista completa.
- Utilización de la memoria como disco ultrarrápido.
- Manejo de errores mejorado.

Memoria intermedia de pista completa

Los disquetes de 5 1/4 pulgadas utilizados en el sistema ejemplificado son de dos caras. Cada cara tiene una cabeza de lectura/escritura separada en la unidad de disco. El controlador físico del disco es lo suficientemente rápido para, si se le ordena, poder leer el contenido completo de una pista de datos de una cara del disquete en una sola revolución de éste.

Los controladores han sido modificados para hacer precisamente esto. La memoria intermedia principal de disco ha sido alargada enormemente para acomodar nueve sectores de 512 bytes.

En el BIOS estándar primitivo, el CP/M estaba configurado para pistas de 18 sectores de 512 bytes. Los datos de cada cabeza en una pista dada estaban colocados "de final a final" con el fin de crear la ilusión de una sola superficie con el doble de datos en ella. Para la memorización de pista, las prestaciones se reducirían si cada lectura requiriese dos revoluciones del disquete, y por ello se ha cambiado en estos BIOS las tablas y el controlador lógico de bajo nivel. Cada superficie está separada, con numeración par de las pistas en la cabeza 0 e impar en la cabeza 1.

El número de pista dado a los controladores de bajo nivel sirve para dos propósitos. El bit menos significativo identifica el número de cabeza. Cuando el número de pista se desplaza un bit a la derecha, el resultado es el número de pista *física* en la cual debe situarse la cabeza.

El algoritmo de desbloqueo se ha modificado también borrando las referencias a los sectores. El programa se refiere ahora únicamente a si el disco y la pista correctos están en la memoria intermedia.

El programa de desbloqueo ya no toma nota cuando el BDOS indica que está escribiendo en un bloque no asignado que utilizó para desviar un sector leído previamente del BIOS estándar. El tamaño de la pista en este BIOS mejorado es mucho mayor que un bloque, por lo que la cuestión es intrascendente; la pista completa debe ser leída previamente para escribir un sector único exclusivamente.

Esta mejora descuella realmente cuando el BDOS está realizando

operaciones de directorio, que incluyen siempre una serie de lecturas secuenciales. El directorio completo puede llevarse a la memoria, actualizado, y reescribirse únicamente en dos revoluciones de disco.

Un punto a observar es lo que se conoce como "escritura diferida". Imagínese un programa instruido para escribir en un sector en la pista 20. Los controladores leerán en la pista 20, copiarán los contenidos del sector designado en la memoria de pista, y volverán al programa *sin* escribir realmente los datos. El programa puede "escribir" en todos los sectores de esta pista sin escribir realmente en el disco. Durante todo este tiempo, estos datos existirán únicamente en memoria y no en el disco, por lo que si ocurre una caída de tensión, se perderán varios miles de bytes de datos. La escritura en el directorio es una excepción. Los controladores escriben siempre físicamente cuando el BDOS indica que está escribiendo en un sector de directorio.

En realidad, el aumento de riesgo es pequeño. La mayoría de los programas están constantemente leyendo y escribiendo ficheros, de forma que la memoria de pista escribirá frecuentemente para leer en otra pista. Cuando los programas terminan, cierran los ficheros de salida. Esto dispara a su vez escrituras del directorio que fuerzan pistas de datos en el disco.

Si la alta seguridad es un requerimiento de la computadora, se puede extender la rutina de centinela para añadir otro temporizador separado. Se puede fijar previamente este cronómetro para, por ejemplo, un retraso de diez segundos cada vez que se escribe en la memoria intermedia de pista, pero no se escribe la memoria en el disco. Cuando la cuenta expira, presentará un indicador que podrá ser comprobado por todos los puntos de entrada del BIOS. Si está a uno, iniciarán una escritura de la memoria de pista en el disco.

Utilización de la memoria como disco ultrarrápido

Como puede verse en la sección precedente, la mejora de rendimiento tiende a estar asociada con el aumento de los requerimientos de memoria. Esto es verdaderamente cierto con un "disco de memoria", denominado normalmente disco RAM o disco M. De hecho, para tener un disco M con capacidad de almacenamiento razonable, la computadora deberá tener por lo menos 128K bytes de memoria adicional.

Como el 8080 o el Z80 sólo pueden direccionar 64K de memoria al mismo tiempo, para tener acceso a esta memoria adicional, alguna parte de la memoria "normal" de la computadora deberá trasladarse desde el espacio de dirección 64K y se deberá conectar la memoria adicional. Esto se conoce como banco de memoria comentada.

La figura 8-7 muestra la organización de memoria que está soportada por los controladores del disco M en el ejemplo.

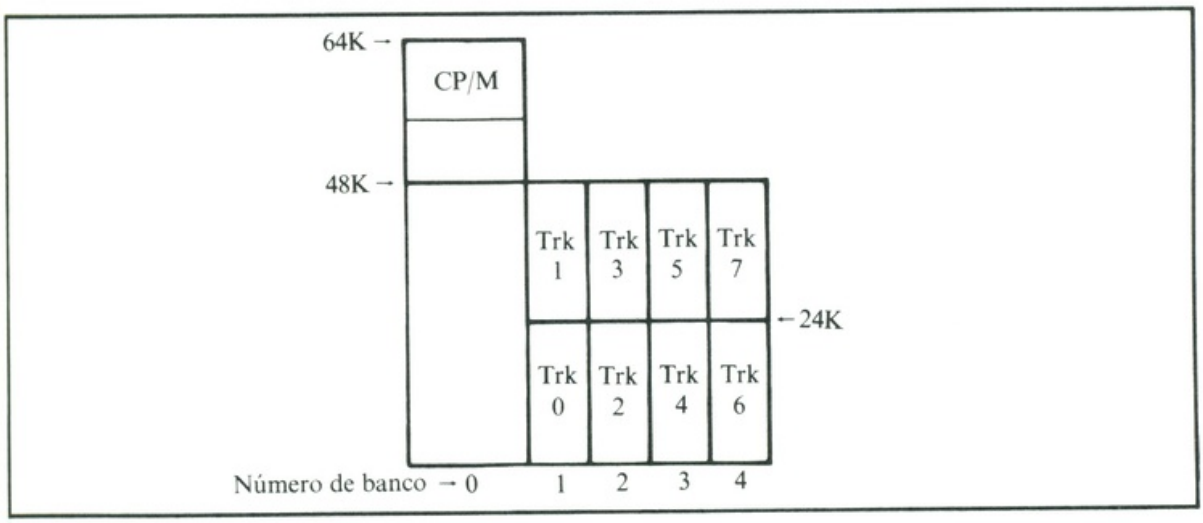


Figura 8-7. Organización de memoria para disco M.

Como puede verse el sistema tiene un total de 256K bytes de RAM, organizados con los 16K superiores, desde el 64K hasta el 48K, que son "comunes" —es decir, están conectados en el espacio de direcciones todo el tiempo—. Los 48K inferiores pueden seleccionarse de los cinco bancos, numerados del 0 al 4. El banco 0 se conecta para las operaciones normales del CP/M.

Los bloques de parámetros del disco M describen un disco con ocho "pistas", numeradas del 0 al 7. El bit menos significativo del número de pista determina si la dirección básica de la pista será 0000H o 6000H. Desplazando el número de pista un bit a la derecha da el número de banco. Cada pista consiste en 192 sectores. Para obtener la dirección relativa de un sector dentro de su "pista", hay que desplazar el número de sector ocho bits a la izquierda y multiplicar entonces por 128.

El disco M está referenciado por el disco lógico M. Se requieren algunas instrucciones para casos especiales para devolver la cabeza de parámetros del disco M especial del SELDSK.

Un problema, afortunadamente de fácil solución, es que la dirección DMA del usuario coexiste en el espacio de direcciones con la imagen del disco M. No existe ninguna forma directa de trasladar datos entre el banco 0 y cualquier otro banco. El disco M utiliza una memoria intermedia, de la parte de memoria común (sobre 48K), que mueve datos en su interior, conmuta bancos y traslada los datos de nuevo. La figura 8-8 muestra un ejemplo de esta secuencia, que se usa cuando se lee del disco M.

Durante la inicialización de arranque en frío, el controlador del disco M comprueba la primerísima dirección de entrada (en el banco 1) para ver si corresponde a una falsa entrada de un fichero denominado "M$Disk". Si esta entrada está presente, el disco M se supone que contiene información

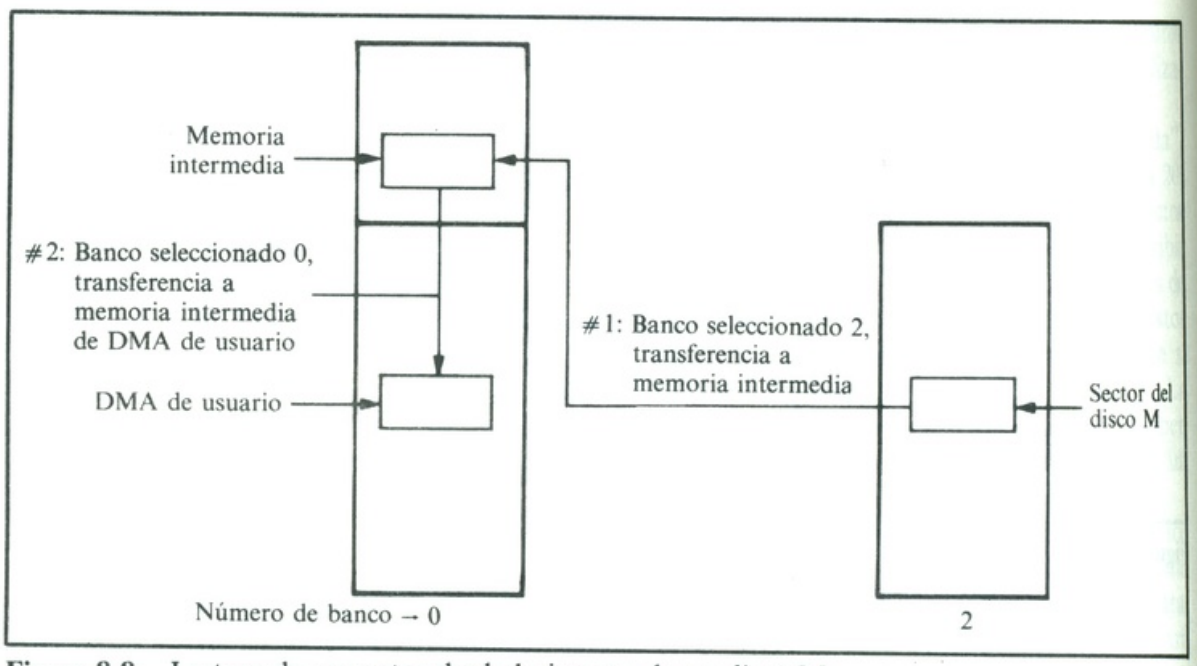


Figura 8-8. Lectura de un sector desde la imagen de un disco M.

válida. Si la entrada está ausente, el programa de inicialización hace esta entrada especial del directorio y llena el resto del directorio con 0E5H, haciéndole aparecer vacío. La entrada falseada hace aparecer que el fichero "M$Disk" está en el usuario 15, marcado en modo sistema y sólo lectura, todo lo cual está pensado para prevenir su borrado accidental.

Añadidos normales para el CP/M

Dos características mostradas en el BIOS mejorado, una en el CCP y otra en el BDOS, requieren cambios en el propio CP/M. Estas características se implementan modificando el CCP y el BDOS para transferir el control al BIOS en puntos específicos, ejecutar algunas instrucciones en el BIOS y volver después al CP/M. Los añadidos pueden hacerse modificando el programa MOVCPM para instalar los cambios de forma permanente. La versión variada del MOVCPM, sin embargo, *deberá* utilizarse con una versión específica del BIOS. Por consiguiente, los añadidos al CP/M "al vuelo" aseguran que no habrá problemas entre el BIOS y el resto del CP/M.

Ambos añadidos se han producido con asistencia de Digital Research.

Ficheros del usuario 0, públicos

El primer cambio permite que los ficheros creados en el área del usuario 0 sean accesibles desde todos los otros números de usuario. Funciona

únicamente con sistemas de disco rígido. En un disco rígido, los números de usuario pueden dividir el disco, pero los programas de utilidad usados frecuentemente han de ser entonces duplicados en cada área de usuario. Permitir que los ficheros en el área de usuario 0 sean públicos, significa que éstos serán accesibles desde todos los otros números de usuario y, por tanto, no necesitan ser copiados en cada área de usuario.

El dispositivo de ficheros públicos altera la forma en que el BDOS realiza la función de búsqueda del próximo, permitiendo el acceso a ficheros declarados en el área de usuario 0 incluso cuando el número real del usuario no sea 0. Sin embargo, el dispositivo es un arma de dos filos —los ficheros del usuario 0 pueden borrarse accidentalmente o dañarse de igual forma que se puede acceder a ellos—. Por consiguiente, los ficheros de usuario 0 deberían ser declarados en modo sistema y de sólo lectura con el fin de protegerlos. Como precaución adicional, los ficheros públicos pueden desconectarse por medio de una señal de control en el bloque de configuración a largo plazo. Esta señal se pone en un estado inicial que inutiliza los ficheros públicos.

Indicación de usuario modificado

Esta modificación hace que el CCP exhiba el número de usuario actual, así como el disco por defecto. Por ejemplo,

```
3B>
```

indica que actualmente se está en el número de usuario 3, con el disco B: como disco por defecto. Además, si se han hecho públicos los ficheros, la indicación estará precedida de la letra "P" para que sirva de recordatorio:

```
P3B>
```

Un BIOS mejorado

El resto de este capítulo es el código fuente en lenguaje ensamblador para el BIOS mejorado que se describe. Esta lista resulta más bien desanimadora, pero vale la pena estudiarla. Los numerosos comentarios se han escrito para hacer más fácil este estudio y se ha recalcado *por qué* y *qué* es lo que hay que hacer.

Como para el BIOS estándar, cada línea se ha numerado de forma que se pueda utilizar el índice funcional de la figura 8-9 para encontrar áreas de interés en el listado. Obsérvese que los números de líneas no son contiguos. Saltan unos cientos al principio de cada sección mayor o subrutina. Esto facilita un número menor de cambios en el listado sin revisión del índice funcional. La lista completa se muestra en la figura 8-10.

Línea de comienzo	Componentes funcionales o rutinas
00001	Comentarios introductorios y equivalencias
00200	Tabla de saltos del BIOS con entradas privadas adicionales
00400	Bloque de configuración a largo plazo
00800	Vector de interrupción
00900	Números de puerto de dispositivos y otras equivalencias
01100	Subrutina Display$Message
01200	Puesta a Enter$CPM
01300	Equivalencias de tabla de dispositivos
01500	Declaraciones de tabla de dispositivos
01700	Inicialización de dispositivos generales
01800	Inicialización de dispositivos específicos
02000	Flujo de bytes de salida
02100	Rutina CONST
02200	Rutina CONIN con proceso de teclas de función
02500	Salida de consola
02700	Rutina CONOUT con proceso de secuencias de escape
02900	AUXIST—Rutina de estado de entrada auxiliar
03000	AUXOST—Rutina de estado de salida auxiliar
03100	AUXIN—Rutina de entrada auxiliar
03200	AUXOUT—Rutina de salida auxiliar
03300	LISTST—Rutina de estado de listado
03400	LIST—Rutina de salida de listado
03500	Elección de usuario requerido—Acción requerida después de error
03600	Salida de mensaje de error
03656	Obtención de estado compuesto desde los dispositivos de salida seleccionados
03800	Salida múltiple de byte a todos los mecanismos de salida
04000	Control del dispositivo de salida preparado lógicamente (protocolo)
04200	Protocolo de proceso ETX/ACK
04400	Tabla de selección de dispositivos desde mapa de bits de redireccionamiento entrada/salida
04600	Obtención de carácter de entrada desde la memoria intermedia de entrada
04800	Comentarios introductorios para los controladores con control de interrupción
04900	Rutina de servicio de interrupción de caracteres
05000	Mecanismo de servicio—Introducción de caracteres en la memoria intermedia de entrada
05300	Obtención de dirección de carácter en memoria intermedia de entrada
05400	Comprobación del carácter de control (no CR, LF, TAB)
05500	Byte de datos de salida
05700	Rutina de estado de entrada
05900	Activación de rutina de temporizador centinela
06000	Rutina de servicio de interrupción de reloj de tiempo real
06200	Rutina de desplazamiento de un bit a la derecha HL
06300	Comentarios introductorios para los controladores de disco de alto nivel
06400	Cabezas de parámetro de disco
06600	Bloques de parámetro de disco
06800	SELDSK—Rutina de selección de disco
07000	SETTRK—Rutina de fijación de pista
07100	SETSEC—Rutina de fijación de sector

Figura 8-9. Indice funcional para listado de la figura 8-10.

```
07200   SETDMA—Rutina de fijación del DMA
07300   Tablas de desplazamiento para traslación de sector
07400   SECTRAN—Rutina de traducción de sector
07500   HOME—Regreso del disco a la pista y al sector 0
07600   Equivalencias para el disco físico y las variables de desbloqueo
07800   READ—Rutina de lectura de sector
07900   WRITE—Rutina de escritura de sector
08000   Código de lectura/escritura normales con algoritmo de desbloqueo
08300   Rutina Move$8—Transfiere la memoria en bloques de 8 bytes
08500   Comentarios introductorios para controladores físicos de disco
08700   Lectura y escritura sin desbloqueo
08900   Controlador de disco M
09100   Rutina de selección de banco de memoria
09200   Lectura/escritura físicas en los discos desbloqueados
09400   Rutinas para manejar errores de disco
09700   Tablas de control de disco para arranque caliente
09800   WBOOT—Rutina de arranque caliente
10000   Servicio de interrupción fantasma
10100   Añadido al CP/M para ficheros públicos y cambios de programa
10300   Obtención de direcciones de bloque de configuración
10400   Direcciones de objetos en bloques de configuración
10500   Bloque de configuración a corto plazo
10700   Nota acerca de por qué los compensadores no inicializados están al final del BIOS
10800   Inicialización de arranque en frío escondida en la memoria intermedia de
        disco seguida por todas las memorias intermedias no inicializadas
```

Figura 8-9. (Continuación.)

```
                00001   ;       Este es un ejemplo esquemático de un BIOS mejorado,
                00010   ;       incluye subrayados fragmentos de BIOS estándar
                00011   ;       mostrados en la figura 6-4 para evitar confusiones
                00012   ;       entre la subestructura de soporte y las mejoras.
                00013   ;       Muchos de los comentarios originales han sido
                00014   ;       abreviados o borrados enteramente.
                00015   ;
                00016   ;
                00017   ;<--    NOTA:   Los números de línea a la izquierda se han incluido
                00018   ;               para proporcionar una referencia entre el programa y
                00019   ;               el texto. Existen discontinuidades deliberadas en los
                00020   ;               números para permitir espacios para expansión.
                00021   ;
3030 =          00022   VERSION         EQU     '00'    ;Equivalencias utilizadas en el mensaje de señal
3230 =          00023   MONTH           EQU     '02'    ; de "funcionando"
3632 =          00024   DAY             EQU     '26'
3338 =          00025   YEAR            EQU     '83'
                00026   ;
                00027   ;*****************************************************************
                00028   ;*    Este BIOS es para un sistema de cálculo con la siguiente   *
                00029   ;*    configuración hardware :                                   *
                00030   ;*                                                               *
                00031   ;*        -- CPU 8080                                            *
                00032   ;*        -- 64K bytes de RAM                                    *
                00033   ;*        -- Controlador de teclado/CRT que transfiere datos como*
                00034   ;*            si fuera un puerto serie [pero sin necesitar un generador*
                00035   ;*            de impulsos para control de velocidad de transmisión (baud rate*
                00036   ;*            generator) o programación de la USART]             *
                00037   ;*        -- Un puerto serie, usado por la impresora o por los dispositivos*
                00038   ;*            de listado, lectura o cinta perforada. Este chip es un*
                00039   ;*            INTEL 8251A con un generador de control-de-velocidad 8253.*
```

Figura 8-10. Listado de BIOS mejorado.

```
                                -- Dos unidades de discos flexibles de 5 1/4", doble cara,
                 ;*                 doble densidad. Estos dispositivos utilizan                 *
                 ;*                 sectores de 512 bytes que son utilizados                    *
                 ;*                 como discos lógicos A: y B:.                                *
                 ;*              -- Dos unidades de disco estándar de 8" (sectores              *
                 ;*                 de 128 bytes) que se usarán como discos lógicos C: y D:.    *
                 ;*                                                                             *
                 ;*                Se utilizan dos controladores de disco inteligentes,         *
                 ;*                 uno para cada tipo de disquete. Estos controladores         *
                 ;*                 acceden directamente a la memoria para leer los detalles    *
                 ;*                 de las operaciones que deben realizar y para leer y escribir *
                 ;*                 datos desde o en los disquetes.                             *
                 ;*****************************************************************************


                 ;        Equivalencias para caracteres en el conjunto de caracteres ASCII
                 ;
0011 =           XON      EQU      11H      ;Permite la transmisión de datos
0013 =           XOFF     EQU      13H      ;Inhibe la transmisión de datos
0003 =           ETX      EQU      03H      ;Fin de transmisión
0006 =           ACK      EQU      06H      ;Contestación
000D =           CR       EQU      0DH      ;Retorno de carro
000A =           LF       EQU      0AH      ;Salto de línea
0009 =           TAB      EQU      09H      ;Tabulador horizontal
0007 =           BELL     EQU      07H      ;Sonido de campana
                 ;
                 ;
                 ;        Las equivalencias para definir el tamaño de memoria, la dirección base
                 ;        y la longitud de los componentes del sistema son:
                 ;
0040 =           Memory$Size      EQU      64       ;Número de Kbytes de RAM
                 ;
                 ;        La longitud del BIOS puede determinarse por inspección.
                 ;        Cambiando el primer carácter de la línea ORG BIOS$Entry, más abajo, por
                 ;        un punto y coma se pondrá ésta como línea de comentario. (Esto hará que
                 ;        el ensamblador comience el BIOS en la posición 0.) Ensamblar el BIOS y
                 ;        marcar en la posición próxima a 100H la dirección que se visualiza en la
                 ;        consola al final del ensamblaje.
                 ;
2500 =           BIOS$Length      EQU      2500H    ;<-- Puesto a un valor aproximado
                                                      ;    para reflejar las mejoras
                 ;
0800 =           CCP$Length       EQU      0800H    ;Constante
0E00 =           BDOS$Length      EQU      0E00H    ;Constante
                 ;
000F =           Overall$Length   EQU      (CCP$Length + BDOS$Length + BIOS$Length + 1023) / 1024
                 ;
C400 =           CCP$Entry        EQU      (Memory$Size - Overall$Length) * 1024
CC06 =           BDOS$Entry       EQU      CCP$Entry + CCP$Length + 6
DA00 =           BIOS$Entry       EQU      CCP$Entry + CCP$Length + BDOS$Length
                 ;
0005 =           BDOS             EQU      0005H    ;Punto de entrada al BDOS (para realizar peticiones
                                                      ;  de puesta a cero del sistema)
                 ;
                 ;#
                 ;        ORG      BIOS$Entry       ;Código de ensamblaje en la dirección del BIOS
                 ;
                 ;        BIOS jump vector
                 ;
0000 C31311              JMP      BOOT     ;Arranque en frío -- entrado desde el cargador de CP/M
                 Warm$Boot$Entry:          ;  Etiquetado de forma que el programa de inicialización
                                           ;  pueda situar la dirección de entrada de arranque caliente
                                           ;  en la posición 0001H y 0002H de la página base
0003 C3750E              JMP      WBOOT    ;Arranque caliente -- entrado saltando a la posición 0000H
                                           ;  Recarga el CCP que puede haber sido reescrito
                                           ;  destructivamente por el programa anterior en el área
                                           ;  de programas transitorios
0006 C32D03              JMP      CONST    ;Estado de consola -- devuelve A = 0FFH si existe un carácter
                                           ;  esperando en la consola
0009 C33A03              JMP      CONIN    ;Entrada de consola -- devuelve en A el siguiente carácter
                                           ;  del teclado de la consola
000C C3D703              JMP      CONOUT   ;Salida de consola -- lanza al dispositivo de consola
                                           ;  el carácter que está en C
000F C3F504              JMP      LIST     ;Salida de impresora -- lanza al dispositivo de impresión
                                           ;  el carácter que está en C
0012 C3CE04              JMP      AUXOUT   ;Salida de perforadora -- lanza al dispositivo lógico de
                                           ;  perforación el carácter que está en C
```

Figura 8-10. (Continuación.)

```
0015 C3A104    00223            JMP     AUXIN   ;Entrada de lectora -- sitúa en A el siguiente carácter de
               00224                            ;  entrada del dispositivo de lectura
0018 C3160A    00225            JMP     HOME    ;Sitúa en la pista 0 al disco seleccionado en curso
001B C36309    00226            JMP     SELDSK  ;Selecciona la unidad de disco especificada en C y devuelve
               00227                            ;  la dirección de los parámetros cabecera del disco
001E C39B09    00228            JMP     SETTRK  ;Fija, a partir del par de registro BC, la pista del disco
               00229                            ;  para la próxima operación de lectura o escritura
0021 C3A109    00230            JMP     SETSEC  ;Fija, a partir del registro A, el sector del disco para la
               00231                            ;  próxima operación de lectura o escritura
0024 C3A809    00232            JMP     SETDMA  ;Fija, a partir del par de registro DE, la dirección
               00233                            ;  de memoria de acceso directo para la próxima operación
               00234                            ;  de lectura o escritura
0027 C3370A    00235            JMP     READ    ;Lee sobre la dirección de DMA la pista y sector del disco
               00236                            ;  seleccionado especificado previamente
002A C34B0A    00237            JMP     WRITE   ;Escribe en el disco seleccionado la pista y sector
               00238                            ;  especificados en el disco selec. desde la dirección de DMA
002D C3D704    00239            JMP     LISTST  ;Devuelve A = 0FFH si el dispositivo de impresión puede
               00240                            ;  aceptar otro carácter de salida
0030 C3100A    00241            JMP     SECTRAN ;Traduce un sector lógico en uno físico
               00242    ;
               00243    ;       Puntos de entrada al BIOS "privados" adicionales
               00244    ;
0033 C38F04    00245            JMP     AUXIST  ;Devuelve A = 0FFH si existe un dato de entrada por el
               00246                            ;  dispositivo lógico auxiliar
0036 C39B04    00247            JMP     AUXOST  ;Devuelve A = 0FFH si el (los) dispositivo auxiliar está
               00248                            ;  preparado lógicamente para aceptar un nuevo byte de salida
0039 C3FA02    00249            JMP     Specific$CIO$Initialization
               00250                            ;Inicializa el dispositivo de caracteres cuyo número de
               00251                            ;  dispositivo está en el registro A
003C C36D08    00252            JMP     Set$Watchdog
               00253                            ;Activa el temporizador centinela para llamar a la dirección
               00254                            ;  especificada en HL después de que transurran BC impulsos
003F C33C0F    00255            JMP     CB$Get$Address
               00256                            ;Dirección de obtención del bloque de configuración
               00257                            ;  Devuelve en HL la dirección de los datos cuyo número
               00258                            ;  de código se especifica en C
               00259    ;
               00400    ;#
               00401    ;       Bloque de configuración a largo plazo
               00402    ;
               00403    Long$Term$CB:
               00404    ;
               00405    ;
               00406    ;       Ficheros públicos (ficheros en el nivel de usuario 0 accesibles
               00407    ;       para todos los números de usuario) disponibles cuando
               00408    ;       este indicador es distinto de cero.
               00409    ;
0042 00        00410    CB$Public$Files:        DB      0       ;Defecto inhibido
               00411    ;
               00412    ;
               00413    ;       El apuntador de entrada forzado es inicializado para apuntar
               00414    ;       a la siguiente cadena de caracteres. Estos son inyectados en el flujo
               00415    ;       de entrada de la consola cuando se levanta el sistema.
               00416    ;
0043 5355424D4900417    CB$Startup:             DB      'SUBMIT STARTUP',LF,0,0,0,0,0,0
               00418    ;
               00419    ;       Redireccionamiento de dispositivos lógicos a físicos
               00420    ;
               00421    ;               Cada dispositivo lógico tiene asociado a él una palabra
               00422    ;               de 16 bits. Cada uno de los bits de la palabra se asigna
               00423    ;               a un dispositivo físico específico. Para entrada únicamente
               00424    ;               uno de los bits puede ponerse a 1 -- la entrada se leerá
               00425    ;               del dispositivo físico correspondiente --. La salida puede
               00426    ;               dirigirse a más de un dispositivo, de manera que más de
               00427    ;               un bit puede estar a uno.
               00428    ;
               00429    ;               Las siguientes equivalencias se usan para indicar dispositivos
               00430    ;               físicos específicos.
               00431    ;
               00432    ;                       1111 11             )
               00433    ;                       5432 1098 7654 3210 )<- Número de dispositivo
0001 =         00434    Device$0        EQU     0000$0000$0000$0001B
0002 =         00435    Device$1        EQU     0000$0000$0000$0010B
0004 =         00436    Device$2        EQU     0000$0000$0000$0100B
               00437    ;
               00438    ;               Las palabras siguientes son comprobadas por el controlador
               00439    ;               de dispositivos lógicos para transferir el control al controlador
```

Figura 8-10. (Continuación.)

```
                        00440   ;                   de dispositivo fisico correspondiente
                        00441   ;
0058 0100               00442   CB$Console$Input:       DW      Device$0
005A 0100               00443   CB$Console$Output:      DW      Device$0
                        00444   ;
005C 0200               00445   CB$Auxiliary$Input:     DW      Device$1
005E 0200               00446   CB$Auxiliary$Output:    DW      Device$1
                        00447   ;
0060 0400               00448   CB$List$Input:          DW      Device$2
0062 0400               00449   CB$List$Output:         DW      Device$2
                        00450   ;
                        00451   ;               La tabla siguiente relaciona bits especificos en las palabras
                        00452   ;               de redireccionamiento con las tablas de dispositivo fisico
                        00453   ;               especifico usadas por los controladores fisicos
                        00454   ;
                        00455   CB$Device$Table$Addresses:
0064 8E02               00456           DW      DT$0
0066 AE02               00457           DW      DT$1
0068 CE02               00458           DW      DT$2
006A 0000000000000459           DW      0,0,0,0,0,0,0,0,0,0,0,0,0       ;No asignado
                        00460   ;
                        00461   ;
                        00462   ;       Flujo de bytes de inicialización de dispositivo
                        00463   ;
                        00464   ;       Este flujo de inicialización se lanza a salida durante
                        00465   ;       la fase de inicialización del dispositivo. Se define
                        00466   ;       en el bloque de configuración a largo plazo para dejar
                        00467   ;       "congelados" sus contenidos desde un arranque hasta
                        00468   ;       el siguiente.
                        00469   ;
                        00470   ;       La dirección de cada flujo está contenida en cada tabla de dispositivo.
                        00471   ;
                        00472   ;       El formato del flujo es:
                        00473   ;
                        00474   ;               DB      xx              ;Número de puerto (termina en 00H)
                        00475   ;               DB      nn              ;Número de bytes a lanzar al puerto
                        00476   ;               DB      vv,vv,vv..      ;Valores a lanzar a salida
                        00477   ;
                        00478   D0$Initialize$Stream:           ;Datos ejemplo para el chip 8251A
0084 ED                 00479           DB      0EDH            ;Número de puerto para el 8251A
0085 06                 00480           DB      6               ;Número de bytes
0086 000000             00481           DB      0,0,0           ;Bytes ficticios para preparación del chip
0089 42                 00482           DB      0100$0010B      ;Puesta a cero y elevación de DTR
008A 6E                 00483           DB      01$10$11$10B    ;1 parada, paridad no, 8 bits/carácter
                        00484                                   ;  factor de división de 16
008B 25                 00485           DB      0010$0101B      ;RTS-alto, disponible Tx/Rx
                        00486                                   ;Datos ejemplo para el chip 8253
008C DF                 00487           DB      0DFH            ;Número de puerto para 8253
008D 01                 00488           DB      1               ;Número de bytes para salida
008E 76                 00489           DB      01$11$011$0B    ;Selección:
                        00490                                   ;       Contador 1
                        00491                                   ;       Cargar primero byte LS
                        00492                                   ;       Modo 3, contador binario
008F DE                 00493           DB      0DEH            ;Número de puerto para el contador
0090 02                 00494           DB      2               ;Número de bytes para salida
                        00495   D0$Baud$Rate$Constant:          ;Etiqueta usada por las utilidades
0091 0700               00496           DW      0007H           ;9.600 baudios (basado en divisor por 16)
0093 00                 00497           DB      0               ;Número de puerto de flujo terminado en 00
                        00498
                        00499   D1$Initialize$Stream:           ;Datos ejemplo para el chip 8251A
0094 DD                 00500           DB      0DDH            ;Número de puerto para el 8251A
0095 06                 00501           DB      6               ;Número de bytes
0096 000000             00502           DB      0,0,0           ;Bytes ficticios para preparación del chip
0099 42                 00503           DB      0100$0010B      ;Puesta a cero y elevación de DTR
009A 6E                 00504           DB      01$10$11$10B    ;1 parada, paridad no, 8 bits/carácter
                        00505                                   ;  factor de división de 16
009B 25                 00506           DB      0010$0101B      ;RTS-alto, disponible Tx/Rx
                        00507
                        00508                                   ;Datos ejemplo para el chip 8253
009C DF                 00509           DB      0DFH            ;Número de puerto para 8253
009D 01                 00510           DB      1               ;Número de bytes para salida
009E B6                 00511           DB      10$11$011$0B    ;Selección:
                        00512                                   ;       Contador 2
                        00513                                   ;       Cargar primero byte LS
                        00514                                   ;       Modo 3, contador binario
009F DE                 00515           DB      0DEH            ;Número de puerto para el contador
00A0 02                 00516           DB      2               ;Número de bytes para salida
```

Figura 8-10. (Continuación.)

```
                        00517   D1$Baud$Rate$Constant:
00A1 3800               00518           DW      0038H                   ;1.200 baudios (basado en divisor por 16)
00A3 00                 00519           DB      0                       ;Número de puerto de flujo terminado en 00
                        00520
                        00521   D2$Initialize$Stream:           ;Datos ejemplo para el chip 8251A
00A4 DD                 00522           DB      0DDH                    ;Número de puerto para el 8251A
00A5 06                 00523           DB      6                       ;Número de bytes
00A6 000000             00524           DB      0,0,0                   ;Bytes ficticios para preparación del chip
00A9 42                 00525           DB      0100$0010B              ;Puesta a cero y elevación de DTR
00AA 6E                 00526           DB      01$10$11$10B            ;1 parada, paridad no, 8 bits/carácter
                        00527                                           ;  factor de división de 16
00AB 25                 00528           DB      0010$0101B              ;RTS-alto, disponible Tx/Rx
                        00529
                        00530                                   ;Datos ejemplo para el chip 8253
00AC DF                 00531           DB      0DFH                    ;Número de puerto para 8253
00AD 01                 00532           DB      1                       ;Número de bytes para salida
00AE F6                 00533           DB      11$11$011$0B            ;Selección:
                        00534                                           ;       Contador 3
                        00535                                           ;       Cargar primero byte LS
                        00536                                           ;       Modo 3, contador binario
00AF DE                 00537           DB      0DEH                    ;Número de puerto para el contador
00B0 02                 00538           DB      2                       ;Número de bytes para salida
                        00539   D2$Baud$Rate$Constant:
00B1 3800               00540           DW      0038H                   ;1.200 baudios (basado en divisor por 16)
00B3 00                 00541           DB      0                       ;Número de puerto de flujo terminado en 00
                        00542
                        00543   ;
                        00544   ;       La tabla siguiente se utiliza para determinar el valor máximo
                        00545   ;       de cada posición de indicación horaria ASCII (excepto los ":").
                        00546   ;       Esta tabla está en el bloque de configuración a largo plazo, de manera
                        00547   ;       que el reloj puede ser puesto permanentemente en formato
                        00548   ;       de 12 ó 24 horas.
                        00549   ;
                        00550   ;       NOTA: La tabla se procesa hacia atrás -- en correspondencia con
                        00551   ;       el tiempo ASCII.
                        00552   ;       Cada carácter representa el valor del correspondiente
                        00553   ;       carácter en "horas" ASCII en el que puede ocurrir un
                        00554   ;       acarreo-y-puesta-a-cero.
                        00555   ;
00B4 00                 00556           DB      0               ;"Terminal"
                        00557   CB$12$24$Clock:
00B5 3334               00558           DB      '34'            ;Cambio de reloj de '23' a '12' horas
00B7 FF                 00559           DB      0FFH            ;Carácter de "salto"
00B8 363A               00560           DB      '6:'            ;Máximo de minutos 59
00BA FF                 00561           DB      0FFH            ;Carácter de "salto"
00BB 363A               00562           DB      '6:'            ;Máximo de segundos es 59
                        00563   Update$Time$End:                ;Utilizado cuando se actualiza "la hora"
                        00564   ;
                        00565   ;
                        00566   ;       Variables para el reloj de tiempo real y para temporizador
                        00567   ;       centinela
                        00568   ;
00BD 3C                 00569   RTC$Ticks$per$Second    DB      60      ;Número de pulsos de reloj
                        00570                                           ;  por segundo transcurrido
00BE 3C                 00571   RTC$Tick$Count          DB      60      ;Cuenta residual entre un segundo
                        00572                                           ;  y el siguiente
00BF 0000               00573   RTC$Watchdog$Count      DW      0       ;Contador de pulsos temporizador centinela
                        00574                                           ;(0 = no activado temporizador centinela)
00C1 0000               00575   RTC$Watchdog$Address    DW      0       ;Dirección a la que será transferido
                        00576                                           ;  el control si la cuenta del temporizador
                        00577                                           ;  centinela llega a 0
                        00578
                        00579   ;
                        00580   ;       Tabla de teclas de función
                        00581   ;
                        00582   ;       Esta tabla consta de una serie de entradas cada una de ellas con la
                        00583   ;       siguiente estructura:
                        00584   ;
                        00585   ;               DB      Segundo carácter de la secuencia emitida por la tecla
                        00586   ;                       de función del terminal
                        00587   ;       (       DB      Tercer carácter de la secuencia -- NOTA: Este campo     )
                        00588   ;       (               no estará presente si el código fuente                  )
                        00589   ;       (               se ha configurado para aceptar sólo dos caracteres      )
                        00590   ;       (               en secuencias de tecla de función.                      )
                        00591   ;       (               NOTA: Ajuste de equivalencias para:                     )
                        00592   ;       (                       Function$key$Length                             )
                        00593   ;       (                       Three$Character$Function                        )
```

Figura 8-10. (Continuación.)

```
                        00594   ;
                        00595   ;                   DB      Una cadena de caracteres es forzada en el flujo de entrada
                        00596   ;                           de consola cuando se pulsa la tecla de función correspondiente.
                        00597   ;                           El último byte de esta cadena debe ser 00H para terminar
                        00598   ;                           la entrada forzada.
                        00599   ;
001B =                  00600   Function$Key$Lead       EQU     1BH     ;Secuencia de señales de tecla de función
0003 =                  00601   Function$Key$Length     EQU     3       ;Número de caracteres en la función
                        00602                                           ;  secuencia de entrada de tecla (NOTA: puede
                        00603                                           ;  ser sólo 3 ó 2 caracteres)
                        00604
                        00605   ;
                        00606                                           ;La lógica asociada con el reconocimiento de
                        00607                                           ;  teclas de función se hace fácilmente
                        00608                                           ;  a partir de la siguiente igualdad
0001 =                  00609   Three$Character$Function        EQU     Function$Key$Length - 2
                        00610                                   ;Three$Character$Function será verdadera (TRUE)
                        00611                                   ;  si la tecla de función emite
                        00612                                   ;  tres caracteres, falsa (FALSE) si emite
                        00613                                   ;  una secuencia de dos caracteres
                        00614
                        00615   ;       Cada entrada en la tabla debe tener la misma longitud, se define:
                        00616   ;
0013 =                  00617   CB$Function$Key$Entry$Size      EQU     16 + 1 + Function$Key$Length - 1
                        00618   ;                                       ^      ^                       ^
                        00619   ;                                       :      :                       :
                        00620   ;               Máxima longitud de la cadena   :       Primer carácter no figura en
                        00621   ;               de sustituto                   :       la entrada de tabla
                        00622   ;                                          Para terminar 00H
                        00623   ;
                        00624   ;       La última entrada en la tabla se marca con un byte 00H
                        00625   ;
                        00626   ;       Los valores ejemplos que se muestran a continuación son para un terminal VT-100
                        00627   ;
                        00628   CB$Function$Key$Table:
                        00629   ;                                       123456789.1234  5  6 7 <- Utilizar para compr. de longitud
00C3 4F5046756E00630            DB      'O','P','Function Key 1',LF,0,0
00D6 4F5146756E00631            DB      'O','Q','Function Key 2',LF,0,0
00E9 4F5246756E00632            DB      'O','R','Function Key 3',LF,0,0
00FC 4F5346756E00633            DB      'O','S','Function Key 4',LF,0,0
                        00634   ;
                        00635   ;                               123456789.1
010F 5B4155702000636            DB      '[','A','Up Arrow',LF,0,0,0,0,0,0,0,0
0122 5B42446F7700637            DB      '[','B','Down Arrow',LF,0,0,0,0,0,0
0135 5B4352696700638            DB      '[','C','Right Arrow',LF,0,0,0,0,0
0148 5B444C656600639            DB      '[','D','Left Arrow',LF,0,0,0,0,0,0
                        00640
015B 000000000000641            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0   ;Entradas disponibles
016E 000000000000642            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0181 000000000000643            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0194 000000000000644            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
01A7 000000000000645            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
01BA 000000000000646            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
01CD 000000000000647            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
01E0 000000000000648            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
01F3 000000000000649            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0206 000000000000650            DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                        00651
0219 FFFF               00652           DB      0FFH,0FFH       ;Terminal para la utilidad que preprograma la
                        00653                                   ;  secuencia de teclas de función
                        00654   ;
                        00655   ;
                        00656   ;       Tabla de control de secuencias de escape de salida de la consola
                        00657   ;
                        00658   ;       Esta tabla se referencia después de que se ha detectado un carácter
                        00659   ;       Function$Key$Lead en la rutina CONOUT. El carácter siguiente a lanzar
                        00660   ;       a salida por consola se compara con el primer byte de cada entrada
                        00661   ;       de tabla de 3 bytes. Si encuentra coincidencia, se transfiere el control
                        00662   ;       a la dirección que sigue al byte coincidente.
                        00663   ;
                        00664   CONOUT$Escape$Table:
021B 74                 00665           DB      't'             ;Lectura de tiempo en curso
021C 4804               00666           DW      CONOUT$Time
021E 64                 00667           DB      'd'             ;Lectura de día en curso
021F 4104               00668           DW      CONOUT$Date
0221 75                 00669           DB      'u'             ;Fijar tiempo en curso
0222 5D04               00670           DW      CONOUT$Set$Time
```

Figura 8-10. (Continuación.)

```
0224 65          00671           DB      'e'             ;Fijar fecha en curso
0225 4E04        00672           DW      CONOUT$Set$Date
                 00673
0227 00          00674           DB      0               ;Terminal
                 00675   ;
                 00676   Long$Term$CB$End:
                 00677   ;
                 00800   ;#
                 00801   ;
                 00802   ;       Vector de interrupciones
                 00803   ;
                 00804   ;       El controlador de interrupciones programable (Intel 8259A) transfiere
                 00805   ;       el control a este punto.
                 00806   ;
                 00807   ;       NOTA: El chip controlador de interrupciones precisa
                 00808   ;             que la tabla de vectores de interrupción empiece en un principio
                 00809   ;             de párrafo. Esto se consigue con la línea ORG siguiente:
0240             00810           ORG     ($ AND OFFE0H) + 20H
                 00811   Interrupt$Vector:
                 00812                           ;Interrupt number
0240 C37808      00813           JMP     RTC$Interrupt           ;0 -- reloj
0243 00          00814           DB      0                       ;Salto de un byte
0244 C3E806      00815           JMP     Character$Interrupt     ;1 -- entrada/salida de carácter
0247 00          00816           DB      0
0248 C3D80E      00817           JMP     Ghost$Interrupt         ;2 -- no utilizada
024B 00          00818           DB      0
024C C3D80E      00819           JMP     Ghost$Interrupt         ;3 -- no utilizada
024F 00          00820           DB      0
0250 C3D80E      00821           JMP     Ghost$Interrupt         ;4 -- no utilizada
0253 00          00822           DB      0
0254 C3D80E      00823           JMP     Ghost$Interrupt         ;5 -- no utilizada
0257 00          00824           DB      0
0258 C3D80E      00825           JMP     Ghost$Interrupt         ;6 -- no utilizada
025B 00          00826           DB      0
025C C3D80E      00827           JMP     Ghost$Interrupt         ;7 -- no utilizada
                 00828   ;
                 00900   ;#
                 00901
                 00902   ;       Números de puerto y otras equivalencias
                 00903   ;
0080 =           00904   CIO$Base$Port     EQU     80H             ;Número de puerto base
                 00905
0080 =           00906   DO$Base$Port      EQU     CIO$Base$Port           ;Dispositivo 0
0080 =           00907   DO$Data$Port      EQU     DO$Base$Port
0081 =           00908   DO$Status$Port    EQU     DO$Base$Port + 1
0082 =           00909   DO$Mode$Port      EQU     DO$Base$Port + 2
0083 =           00910   DO$Command$Port   EQU     DO$Base$Port + 3
                 00911   ;
                 00912
0084 =           00913   D1$Base$Port      EQU     CIO$Base$Port + 4       ;Dispositivo 1
0084 =           00914   D1$Data$Port      EQU     D1$Base$Port
0085 =           00915   D1$Status$Port    EQU     D1$Base$Port + 1
0086 =           00916   D1$Mode$Port      EQU     D1$Base$Port + 2
0087 =           00917   D1$Command$Port   EQU     D1$Base$Port + 3
                 00918
0088 =           00919   D2$Base$Port      EQU     CIO$Base$Port + 8       ;Dispositivo 2
0088 =           00920   D2$Data$Port      EQU     D2$Base$Port
0089 =           00921   D2$Status$Port    EQU     D2$Base$Port + 1
008A =           00922   D2$Mode$Port      EQU     D2$Base$Port + 2
008B =           00923   D2$Command$Port   EQU     D2$Base$Port + 3
                 00924
004E =           00925   D$Mode$Value$1    EQU     01$00$11$10B
                 00926                             ;1 bit de parada, paridad no
                 00927                             ;8 bits, asinc., factor 16
003C =           00928   D$Mode$Value$2    EQU     00$11$1100B
                 00929                             ;Tx/Rx en el reloj interno
                 00930                             ;9.600 baudios
0027 =           00931   D$Command$Value   EQU     00$100111B
                 00932                             ;Modo normal
                 00933                             ;Tx/Rx disponible
                 00934                             ;Activar RTS y DTR
0038 =           00935   D$Error           EQU     0011$1000B
0037 =           00936   D$Error$Reset     EQU     00$110111B
                 00937                             ;Lo mismo que valor de orden más puesta a cero de error
0001 =           00938   D$Output$Ready    EQU     0000$0001B
0002 =           00939   D$Input$Ready     EQU     0000$0010B
0080 =           00940   D$DTR$High        EQU     1000$0000B      ;Nota: esto es realmente el pin DSR
```

Figura 8-10. (Continuación.)

```
                        00941                                   ;  (data-set-ready) del
                        00942                                   ;  chip. Está conectado al
                        00943                                   ;  pin DTR
0027 =                  00944   D$Raise$RTS      EQU     00$1$00111B     ;Elevar RTS, Tx/Rx preparado
0007 =                  00945   D$Drop$RTS       EQU     00$0$00111B     ;Bajar RTS, Tx/Rx preparado
                        00946   ;
                        00947   ;
                        00948   ;       Puerto controlador de interrupciones (Intel 8259A)
                        00949   ;
                        00950   ;       Nota : Estas equivalencias se sitúan en este lugar siguiendo
                        00951   ;              a la definición de vector de interrupciones
                        00952   ;              y así evitar errores 'P' (fase) en el ASM.
                        00953   ;
00D9 =                  00954   IC$OCW1$Port     EQU     0D9H    ;Palabra de control operacional 1
00D8 =                  00955   IC$OCW2$Port     EQU     0D8H    ;Palabra de control operacional 2
00D8 =                  00956   IC$OCW3$Port     EQU     0D8H    ;Palabra de control operacional 3
00D8 =                  00957   IC$ICW1$Port     EQU     0D8H    ;Palabra de control de inicialización 1
00D9 =                  00958   IC$ICW2$Port     EQU     0D9H    ;Palabra de control de inicialización 2
                        00959   ;
0020 =                  00960   IC$EOI           EQU     20H     ;Final no especifico de interrupción
                        00961   ;
0056 =                  00962   IC$ICW1          EQU     (Interrupt$Vector AND 1110$0000B) + 000$10110B
                        00963                                   ;Fijación de los bits A7-A5 de la dirección
                        00964                                   ;  del vector de interrupciones más:
                        00965                                   ;       Activación por flancos
                        00966                                   ;       Intervalo de 4 bytes
                        00967                                   ;       8259 único en el sistema
                        00968                                   ;       No necesita ICW4
0002 =                  00969   IC$ICW2          EQU     Interrupt$Vector SHR 3
                        00970                                   ;Bits de dirección A15-A8 de la dirección del vector
                        00971                                   ;  de interrupciones. Nótese que el vector
                        00972                                   ;  de interrupciones es la primera estructura
                        00973                                   ;  en el bloque de configuración a largo plazo
                        00974
00FC =                  00975   IC$OCW1          EQU     1111$1100B      ;Máscara de interrupciones
                        00976                                   ;Interrupción 0 (reloj) activada
                        00977                                   ;Interrupción 1 (entrada de carácter) activada
                        00978   ;
                        01100   ;#
                        01101   ;
                        01102   ;
                        01103   Display$Message:                ;Visualización en la consola del mensaje especificado
                        01104                                   ;En entrada, HL apunta al flujo de bytes a sacar.
                        01105                                   ;Un byte 00 termina el mensaje.
025F 7E                 01106           MOV     A,M             ;Obtención byte siguiente de mensaje
0260 B7                 01107           ORA     A               ;Comprobación de si es un terminal
0261 C8                 01108           RZ                      ;Si, retorno al llamador
0262 4F                 01109           MOV     C,A             ;Preparación para salida
0263 E5                 01110           PUSH    H               ;Almacenamiento de apuntador a mensaje
0264 CDD703             01111           CALL    CONOUT          ;Ida a la rutina principal de salida por consola
0267 E1                 01112           POP     H               ;Recuperación apuntador a mensaje
0268 23                 01113           INX     H               ;Desplazamiento a byte siguiente de mensaje
0269 C35F02             01114           JMP     Display$Message ;Bucle hasta completar salida de mensaje
                        01115   ;
                        01200   ;#
                        01201   ;
                        01202   Enter$CPM:          ;Esta rutina se carga desde los programas de arranque en frío y
                        01203                       ;  en caliente. Fija las instrucciones JMP en la
                        01204                       ;  página base y la dirección de DMA de la
                        01205                       ;  entrada/salida de las unidades de disco.
                        01206   ;
026C 3EC3               01207           MVI     A,JMP           ;Obtención del código máquina para los JMP
026E 320000             01208           STA     0000H           ;Fijación de los JMP en la posición 0000H
0271 320500             01209           STA     0005H           ;  y en la posición 0005H
                        01210   ;
0274 210300             01211           LXI     H,Warm$Boot$Entry       ;Obtención de la direc. del vector del BIOS
0277 220100             01212           SHLD    0001H           ;Puesta de la dirección en la posición 0001H
                        01213
027A 2106CC             01214           LXI     H,BDOS$Entry    ;Obtención de la direc. del punto de entrada al BDOS
027D 220600             01215           SHLD    6               ;Puesta de la dirección en la posición 0005H
                        01216   ;
0280 018000             01217           LXI     B,80H           ;Fijación de direc. E/S de disco a su valor por defecto
0283 CDA809             01218           CALL    SETDMA          ;Utilización de la rutina BIOS normal
                        01219   ;
0286 FB                 01220           EI                      ;Asegurar activación de interrupciones
0287 3A0400             01221           LDA     Default$Disk    ;Manejo del disco por defecto en curso al
028A 4F                 01222           MOV     C,A             ;  procesador de comandos de consola.
```

Figura 8-10. (Continuación.)

```
028B C300C4     01223           JMP     CCP$Entry       ;Transferencia al CCP
                01224   ;
                01300   ;#
                01301   ;
                01302   ;       Equivalencias de tabla de dispositivo
                01303   ;       Los controladores utilizan una tabla de dispositivo
                01304   ;       para cada dispositivo físico al que sirven. Las equivalencias
                01305   ;       que siguen se utilizan para acceder a diferentes campos
                01306   ;       en el interior de la tabla.
                01307   ;
                01308   ;                               Números de puerto y bits de estado
0000 =          01309   DT$Status$Port          EQU     0       ;Número de puerto de estado de dispositivo
0001 =          01310   DT$Data$Port            EQU     DT$Status$Port+1
                01311                                   ;Número de puerto de dispositivo de datos
0002 =          01312   DT$Output$Ready         EQU     DT$DataPort+1
                01313                                   ;Máscara de estado de salida preparada
0003 =          01314   DT$Input$Ready          EQU     DT$Output$Ready+1
                01315                                   ;Máscara de estado de entrada preparada
0004 =          01316   DT$DTR$Ready           EQU     DT$Input$Ready+1
                01317                                   ;DTR preparado para enviar máscara
0005 =          01318   DT$Reset$Int$Port      EQU     DT$DTR$Ready+1
                01319                                   ;Número de puerto preparado para enviar
                01320                                   ;  máscara cero de interrupción
0006 =          01321   DT$Reset$Int$Value     EQU     DT$Reset$Int$Port+1
                01322                                   ;Valor de salida puesta a cero interrupción
0007 =          01323   DT$Detect$Error$Port   EQU     DT$Reset$Int$Value+1
                01324                                   ;Número de puerto para detección de error
0008 =          01325   DT$Detect$Error$Value  EQU     DT$Detect$Error$Port+1
                01326                                   ;Máscara para detec. de error (paridad, etc.)
0009 =          01327   DT$Reset$Error$Port     EQU     DT$Detect$Error$Value+1
                01328                                   ;Salida a puerto para puesta a cero de error
000A =          01329   DT$Reset$Error$Value    EQU     DT$Reset$Error$Port+1
                01330                                   ;Valor de salida para puesta a cero de error
000B =          01331   DT$RTS$Control$Port     EQU     DT$Reset$Error$Value+1
                01332                                   ;Puerto de control para RTS bajo
000C =          01333   DT$Drop$RTS$Value       EQU     DT$RTS$Control$Port+1
                01334                                   ;Valor para bajar RTS, al lanzarlo
000D =          01335   DT$Raise$RTS$Value      EQU     DT$Drop$RTS$Value+1
                01336                                   ;Valor para elevar RTS, al lanzarlo
                01337   ;
                01338   ;               Estado de dispositivo lógico (incl. protocolos)
000E =          01339   DT$Status               EQU     DT$Raise$RTS$Value+1
                01340                                           ;Bits de estado
0001 =          01341   DT$Output$Suspend       EQU     0000$0001B      ;Salida suspendida pendiente de la
                01342                                           ;  acción del protocolo
0002 =          01343   DT$Input$Suspend        EQU     0000$0010B      ;Salida suspendida hasta que se
                01344                                           ;  vacíe la memoria intermedia
0004 =          01345   DT$Output$DTR           EQU     0000$0100B      ;La sal. utiliza DTR-alto para enviar
0008 =          01346   DT$Output$Xon           EQU     0000$1000B      ;La salida utiliza XON/XOFF
0010 =          01347   DT$Output$Etx           EQU     0001$0000B      ;La salida utiliza ETX/ACK
0020 =          01348   DT$Output$Timeout       EQU     0010$0000B      ;La salida utiliza límite de tiempo
0040 =          01349   DT$Input$RTS            EQU     0100$0000B      ;La sal. utiliza TRS-alto para recibir
0080 =          01350   DT$Input$Xon            EQU     1000$0000B      ;La salida utiliza XON/XOFF
                01351   ;
000F =          01352   DT$Status$2             EQU     DT$Status+1     ;Byte de estado secundario
0001 =          01353   DT$Fake$Typeahead      EQU     0000$0001B      ;Petición de Input$Status para
                01354                                                   ; devolver "Datos preparados"
                01355                                                   ; cuando hay carácter de control en
                01356                                                   ; la memoria intermedia de entrada
                01357   ;
0010 =          01358   DT$Etx$Count            EQU     DT$Status$2+1
                01359                                   ;Número caracteres enviados con protocolo Etx
0012 =          01360   DT$Etx$Message$Length   EQU     DT$Etx$Count+2
                01361                                   ;Longitud del mensaje especificado
                01362   ;
                01363   ;                       Input buffer values
0014 =          01364   DT$Buffer$Base          EQU     DT$Etx$Message$Length+2
                01365                                   ;Dirección de la mem. int. de entrada
0016 =          01366   DT$Put$Offset           EQU     DT$Buffer$Base+2
                01367                                   ;Desplaz. para puesta de carác. en mem. int.
0017 =          01368   DT$Get$Offset           EQU     DT$Put$Offset+1
                01369                                   ;Desplaz. para obtención de carác. en mem. int.
0018 =          01370   DT$Buffer$Length$Mask   EQU     DT$Get$Offset+1
                01371                                   ;Longitud de la memoria intermedia - 1
                01372                                   ;Nota: La longitud de la mem. int. debe ser
                01373                                   ;  siempre potencia de 2: 32, 64, 128
```

Figura 8-10. (Continuación.)

```
                        01374                                   ;Esta máscara se convierte entonces en:
                        01375                                   ;  32 ->  31 (0001$1111B)
                        01376                                   ;  64 ->  63 (0011$1111B)
                        01377                                   ; 128 -> 127 (0111$1111B)
                        01378                                   ;Después de la aplicación de la función
                        01379                                   ;  obtención/puesta el despl. se ha incrementado,
                        01380                                   ;  se opera (AND) con la máscara para ponerlo a
                        01381                                   ;  cero cuando se alcance el final de la mem. int.
0019 =                  01382   DT$Character$Count      EQU     DT$Buffer$Length$Mask+1
                        01383                                   ;Cuenta del número de caracteres en curso
                        01384                                   ;  en la memoria intermedia
001A =                  01385   DT$Stop$Input$Count     EQU     DT$Character$Count+1
                        01386                                   ;Parar entrada cuando el contador alcance
                        01387                                   ;  este valor
001B =                  01388   DT$Resume$Input$Count   EQU     DT$Stop$Input$Count+1
                        01389                                   ;Reinicio de la entrada cuando el contador
                        01390                                   ;  alcanza este valor
001C =                  01391   DT$Control$Count        EQU     DT$Resume$Input$Count+1
                        01392                                   ;Cuenta del número de caracteres de control
                        01393                                   ;  en la memoria intermedia
001D =                  01394   DT$Function$Delay       EQU     DT$Control$Count+1
                        01395                                   ;Número de pulsos de reloj a retrasar para
                        01396                                   ;  permitir la llegada de todos los caracteres
                        01397                                   ;  después de pulsar la tecla de función
001E =                  01398   DT$Initialize$Stream   EQU     DT$Function$Delay+1
                        01399                                   ;Dirección del flujo de bytes necesario
                        01400                                   ;  para inicializar
                        01401
                        01500   ;#
                        01501   ;
                        01502   ;       Tablas de dispositivo
                        01503   ;
                        01504   DT$0:
028E 81                 01505           DB      D0$Status$Port  ;Puerto de estado (chip 8251A)
028F 80                 01506           DB      D0$Data$Port    ;Puerto de datos
0290 01                 01507           DB      D$Output$Ready  ;Salida de datos preparada
0291 02                 01508           DB      D$Input$Ready   ;Entrada de datos preparada
0292 80                 01509           DB      D$DTR$High      ;DTR preparado para mandar
0293 D8                 01510           DB      IC$OCW2$Port    ;Puesta a 0 puerto de int. (OOH es puerto no utilizado)
0294 20                 01511           DB      IC$EOI          ;Puesta a 0 valor de int. (EOI no especifica)
0295 81                 01512           DB      D0$Status$Port  ;Detección de puerto error
0296 38                 01513           DB      D$Error         ;Máscara, estado del proc., esc. encima, error paridad
0297 83                 01514           DB      D0$Command$Port ;Puesta a cero puerto error
0298 37                 01515           DB      D$Error$Reset   ;Puesta a cero: RTS-alto, Tx/Rx activo
0299 83                 01516           DB      D0$Command$Port ;Bajada/elevación puerto RTS
029A 07                 01517           DB      D$Drop$RTS      ;Bajar valor RTS (mantener Tx/Rx activo)
029B 27                 01518           DB      D$Raise$RTS     ;Elevar valor RTS (mantener Tx/Rx activo)
029C C0                 01519           DB      DT$Input$Xon + DT$Input$RTS     ;Protocolo y estado
029D 00                 01520           DB      0               ;Estado #2
029E 0004               01521           DW      1024            ;Etx/Ack contador de mensaje
02A0 0004               01522           DW      1024            ;Etx/Ack longitud de mensaje
02A2 2422               01523           DW      D0$Buffer       ;Memoria intermedia de entrada
02A4 00                 01524           DB      0               ;Puesta del desplazamiento en la memoria intermedia
02A5 00                 01525           DB      0               ;Obtención del desplazamiento en la mem. int.
02A6 1F                 01526           DB      D0$Buffer$Length -1  ;Máscara de longitud memoria intermedia
02A7 00                 01527           DB      0               ;Cuenta de caracteres en la memoria intermedia
02A8 1B                 01528           DB      D0$Buffer$Length- 5 ;Parar ent. cuando contador llega a este valor
02A9 10                 01529           DB      D0$Buffer$Length/ 2 ;Reanudar ent. cuando contador llega a este valor
02AA 00                 01530           DB      0               ;Cuenta del núm. de carac. de control en la mem. int.
02AB 06                 01531           DB      6               ;Número 16,66 pulsos para permitir la llegada
                        01532                                   ;  de la secuencia de la tecla de función
02AC 8400               01533           DW      D0$Initialize$Stream    ;Dirección del flujo de inicialización
                        01534   ;
                        01535   DT$1:
02AE 85                 01536           DB      D1$Status$Port  ;Puerto de estado (chip 8251A)
02AF 84                 01537           DB      D1$Data$Port    ;Puerto de datos
02B0 01                 01538           DB      D$Output$Ready  ;Salida de datos preparada
02B1 02                 01539           DB      D$Input$Ready   ;Entrada de datos preparada
02B2 80                 01540           DB      D$DTR$High      ;DTR preparado para mandar
02B3 D8                 01541           DB      IC$OCW2$Port    ;Puesta a 0 puerto de int. (OOH es puerto no utilizado)
02B4 20                 01542           DB      IC$EOI          ;Puesta a 0 valor de int. (EOI no especifica)
02B5 85                 01543           DB      D1$Status$Port  ;Detección de puerto error
02B6 38                 01544           DB      D$Error         ;Máscara, estado del proc., esc. encima, error paridad
02B7 87                 01545           DB      D1$Command$Port ;Puesta a cero puerto error
02B8 37                 01546           DB      D$Error$Reset   ;Puesta a cero: RTS-alto, Tx/Rx activo
02B9 87                 01547           DB      D1$Command$Port ;Bajada/elevación puerto RTS
02BA 07                 01548           DB      D$Drop$RTS      ;Bajar valor RTS (mantener Tx/Rx activo)
```

Figura 8-10. (Continuación.)

```
02BB 27          01549              DB      D$Raise$RTS       ;Elevar valor RTS (mantener Tx/Rx activo)
02BC C0          01550              DB      DT$Input$Xon + DT$Input$RTS    ;Protocolo y estado
02BD 00          01551              DB      0                 ;Estado #2
02BE 0004        01552              DW      1024              ;Etx/Ack contador de mensaje
02C0 0004        01553              DW      1024              ;Etx/Ack longitud de mensaje
02C2 4422        01554              DW      D1$Buffer         ;Memoria intermedia de entrada
02C4 00          01555              DB      0                 ;Puesta del desplazamiento en la memoria intermedia
02C5 00          01556              DB      0                 ;Obtención del desplazamiento en la mem. int.
02C6 1F          01557              DB      D1$Buffer$Length -1  ;Máscara de longitud memoria intermedia
02C7 00          01558              DB      0                 ;Cuenta de caracteres en la memoria intermedia
02C8 1B          01559              DB      D1$Buffer$Length- 5 ;Parar ent. cuando contador llega a este valor
02C9 10          01560              DB      D1$Buffer$Length/ 2 ;Reanudar ent. cuando contador llega a este valor
02CA 00          01561              DB      0                 ;Cuenta del núm. de carac. de control en la mem. int.
02CB 06          01562              DB      6                 ;Número de 16,6 pulsos para permitir la llegada
                 01563                                        ;  de la secuencia de la tecla de función
02CC 9400        01564              DW      D1$Initialize$Stream    ;Dirección del flujo de inicialización
                 01565    ;
                 01566    ;
                 01567    DT$2:
02CE 89          01568              DB      D2$Status$Port    ;Puerto de estado (chip 8251A)
02CF 88          01569              DB      D2$Data$Port      ;Puerto de datos
02D0 01          01570              DB      D$Output$Ready    ;Salida de datos preparada
02D1 02          01571              DB      D$Input$Ready     ;Entrada de datos preparada
02D2 80          01572              DB      D$DTR$High        ;DTR preparado para mandar
02D3 D8          01573              DB      IC$OCW2$Port      ;Puesta a 0 puerto de int. (00H es puerto no utilizado)
02D4 20          01574              DB      IC$EOI            ;Puesta a 0 valor de int. (EOI no especifica)
02D5 89          01575              DB      D2$Status$Port    ;Detección de puerto error
02D6 38          01576              DB      D$Error           ;Máscara, estado del proc., esc. encima, error paridad
02D7 8B          01577              DB      D2$Command$Port   ;Puesta a cero puerto error
02D8 37          01578              DB      D$Error$Reset     ;Puesta a cero: RTS-alto, Tx/Rx activo
02D9 8B          01579              DB      D2$Command$Port   ;Bajada/elevación puerto RTS
02DA 07          01580              DB      D$Drop$RTS        ;Bajar valor RTS (mantener Tx/Rx activo)
02DB 27          01581              DB      D$Raise$RTS       ;Elevar valor RTS (mantener Tx/Rx activo)
02DC C0          01582              DB      DT$Input$Xon + DT$Input$RTS    ;Protocolo y estado
02DD 00          01583              DB      0                 ;Estado #2
02DE 0004        01584              DW      1024              ;Etx/Ack contador de mensaje
02E0 0004        01585              DW      1024              ;Etx/Ack longitud de mensaje
02E2 6422        01586              DW      D2$Buffer         ;Memoria intermedia de entrada
02E4 00          01587              DB      0                 ;Puesta del desplazamiento en la memoria intermedia
02E5 00          01588              DB      0                 ;Obtención del desplazamiento en la mem. int.
02E6 1F          01589              DB      D2$Buffer$Length -1  ;Máscara de longitud memoria intermedia
02E7 00          01590              DB      0                 ;Cuenta de caracteres en la memoria intermedia
02E8 1B          01591              DB      D2$Buffer$Length- 5 ;Parar ent. cuando contador llega a este valor
02E9 10          01592              DB      D2$Buffer$Length/ 2 ;Reanudar ent. cuando contador llega a este valor
02EA 00          01593              DB      0                 ;Cuenta del núm. de carac. de control en la mem. int.
02EB 06          01594              DB      6                 ;Número de 16,66 pulsos para permitir la llegada
                 01595                                        ;  de la secuencia de la tecla de función
02EC A400        01596              DW      D2$Initialize$Stream    ;Dirección del flujo de inicialización
                 01597    ;
                 01700    ;#
                 01701    ;        Inicialización general de dispositivo de E/S de caracteres
                 01702    ;
                 01703    ;        Esta rutina será llamada desde el programa principal
                 01704    ;        de inicialización de CP/M.
                 01705    ;
                 01706    ;        Hace varias llamadas a la rutina específica de inicialización
                 01707    ;        de dispositivos de E/S.
                 01708    ;
                 01709    General$CIO$Initialization:
02EE AF          01710              XRA     A                 ;Fijación número de dispositivo (utilizado para
                 01711                                       ;  acceder a la tabla de direcciones de tablas de
                 01712                                       ;  dirección del bloque de configuración)
02EF 4F          01713              MOV     C,A               ;Emparejamiento con interfaz llamable externamente
                 01714    GCI$Next$Device:
02F0 CDFA02      01715              CALL    Specific$CIO$Initialization    ;Inicialización del dispositivo
02F3 3C          01716              INR     A                 ;Mover a dispositivo siguiente
02F4 FE10        01717              CPI     16                ;Comprobación de inicialización de todos los
02F6 C8          01718              RZ                        ;  dispositivos (0 - 15) posibles
02F7 C3F002      01719              JMP     GCI$Next$Device
                 01720    ;
                 01800    ;#
                 01801    ;
                 01802    ;        Inicialización de dispositivos específicos de E/S de caracteres
                 01803    ;
                 01804    ;        Esta rutina lanza a la salida valores específicos de bytes a los puertos
                 01805    ;        especificados controlados por los flujos de inicialización del bloque de
                 01806    ;        configuración. Cada tabla de dispositivo contiene un apuntador
```

Figura 8-10. (Continuación.)

```
                        01807   ;       a estos flujos. La tabla de dispositivos se relaciona
                        01808   ;       de acuerdo con el número de dispositivo -- es un parámetro
                        01809   ;       de entrada para esta rutina.
                        01810   ;       La rutina puede ser llamada desde la rutina anterior de inicialización
                        01811   ;       general o directamente por una llamada a BIOS desde
                        01812   ;       una utilidad del sistema ejecutándose en la TPA.
                        01813   ;
                        01814   ;       Parámetros de entrada
                        01815   ;
                        01816   ;               C = número de dispositivo
                        01817   ;
                        01818   ;       Parámetros de salida
                        01819   ;
                        01820   ;               A = número de dispositivo (preservado)
                        01821   ;
                        01822   ;============================
                        01823   Specific$CIO$Initialization:            ;<=== Punto de entrada al BIOS (privado)
                        01824   ;============================
02FA 79                 01825           MOV     A,C                     ;Obtención número de dispositivo
02FB F5                 01826           PUSH    PSW                     ;Preservar número de dispositivo
02FC 87                 01827           ADD     A                       ;Componer número de dispositivo en apuntador
02FD 4F                 01828           MOV     C,A
02FE 0600               01829           MVI     B,0                     ;Componer en una palabra
0300 216400             01830           LXI     H,CB$Device$Table$Addresses  ;Obtención tabla base
0303 09                 01831           DAD     B                       ;HL -> dirección tabla de dispositivo
0304 5E                 01832           MOV     E,M                     ;Obtención byte LS
0305 23                 01833           INX     H
0306 56                 01834           MOV     D,M                     ;Obtención byte MS DE -> tabla de dispositivo
                        01835
0307 7A                 01836           MOV     A,D                     ;Comprobación de dirección tabla dispositivo = 0
0308 B3                 01837           ORA     E
0309 CA1703             01838           JZ      SCI$Exit                ;Sí, tabla de dispositivo inexistente
                        01839
030C 211E00             01840           LXI     H,DT$Initialize$Stream
030F 19                 01841           DAD     D                       ;HL -> dirección flujo de inicialización
0310 5E                 01842           MOV     E,M                     ;Obtención byte LS
0311 23                 01843           INX     H
0312 56                 01844           MOV     D,M                     ;Obtención byte MS
0313 EB                 01845           XCHG                            ;HL -> flujo de inicialización
0314 CD1903             01846           CALL    Output$Byte$Stream      ;Salida flujo de bytes
                        01847                                           ; a diversos puertos
                        01848   ;
                        01849   SCI$Exit:
0317 F1                 01850           POP     PSW                     ;Recuperar número de dispositivo de usuario en C
0318 C9                 01851           RET
                        01852   ;
                        02000   ;*
                        02001   ;       Salida de flujo de bytes
                        02002   ;
                        02003   ;       Esta rutina lanza a salida bytes de inicialización a números
                        02004   ;       de puerto. El flujo de bytes tiene el formato siguiente:
                        02005   ;
                        02006   ;               DB      ppH     Número de puerto
                        02007   ;               DB      nn      Número de bytes para salida
                        02008   ;               DB      vvH,vvH...      Bytes para salida
                        02009   ;               :
                        02010   ;               :       Repetición
                        02011   ;               :
                        02012   ;               DB      00H     Número de puerto de 0
                        02013   ;
                        02014   ;       Parámetros de entrada
                        02015   ;
                        02016   ;               HL -> flujo de bytes
                        02017   ;
                        02018   Output$Byte$Stream:
                        02019   OBS$Loop:
0319 7E                 02020           MOV     A,M                     ;Obtención número de puerto
031A B7                 02021           ORA     A                       ;Comprobación de 00H (terminal)
031B C8                 02022           RZ                              ;Salida si final de flujo
031C 322503             02023           STA     OBS$Port                ;Almacenar en número de puerto
031F 23                 02024           INX     H                       ;HL -> cuenta de bytes
0320 4E                 02025           MOV     C,M                     ;Obtención cuenta
0321 23                 02026           INX     H                       ;HL -> primer byte de inicialización
                        02027   ;
                        02028   OBS$Next$Byte:
0322 7E                 02029           MOV     A,M                     ;Obtención byte siguiente
0323 23                 02030           INX     H                       ;HL -> byte de datos siguiente (o número de puerto)
```

Figura 8-10. (Continuación.)

```
                02031
0324 D3         02032           DB      OUT
                02033   OBS$Port:
0325 00         02034           DB      0               ;<- Puesta a 1 en instrucción anterior
0326 0D         02035           DCR     C               ;Decrementar contador de bytes
0327 C22203     02036           JNZ     OBS$Next$Byte   ;Salida byte datos siguiente
032A C31903     02037           JMP     OBS$Loop        ;Vuelta atrás para siguiente número de puerto
                02038   ;
                02100   ;*
                02101   ;       CONST - Estado de consola
                02102   ;
                02103   ;       Esta rutina comprueba el apuntador de entrada forzada
                02104   ;       y la cuenta de caracteres para la memoria intermedia
                02105   ;       apropiada. El registro A se activa para indicar cuándo
                02106   ;       existen datos a la espera.
                02107   ;
                02108   ;       Parámetros de entrada: ninguno.
                02109   ;
                02110   ;       Parámetros de salida
                02111   ;
                02112   ;               A = 000H si no hay datos esperando
                02113   ;               A = 0FFH si hay datos esperando
                02114   ;
                02115   ;===========================
                02116   CONST:                                  ;<=== Punto de entrada al BIOS (estándar)
                02117   ;===========================
032D 2A5800     02118           LHLD    CB$Console$Input        ;Obtención palabra de redireccionamiento
0330 116400     02119           LXI     D,CB$Device$Table$Addresses
0333 CD6F06     02120           CALL    Select$Device$Table     ;Obtención tabla de dispositivos
0336 C34708     02121           JMP     Get$Input$Status        ;Obtención estado de dispositivo de entrada
                02122                                           ;  y retorno a quien efectúa la llamada
                02200   ;*
                02201   ;
                02202   ;       CONIN -- entrada de consola
                02203   ;
                02204   ;       Esta rutina devuelve el carácter siguiente del flujo de entrada de
                02205   ;       consola. Dependiendo de las circunstancias puede tratarse de un carácter
                02206   ;       de la memoria intermedia de entrada de consola o de una cadena de
                02207   ;       caracteres almacenada previamente para ser "forzada" en el flujo de
                02208   ;       entrada por las rutinas de ejecución o de inicialización del sistema. La
                02209   ;       "entrada forzada" puede provenir de una cadena de caracteres almacenada
                02210   ;       previamente en memoria. Se utiliza para inyectar la fecha y hora en
                02211   ;       curso o una cadena asociada con una tecla de función en el flujo de
                02212   ;       consola. Al levantar el sistema, se fuerza la cadena "SUBMIT STARUP" en
                02213   ;       el flujo de entrada de consola para proveer el mecanismo.
                02214   ;
                02215   ;       La entrada normal (no forzada) de un dispositivo físico cualquiera se
                02216   ;       especifica en la palabra de redireccionamiento de entrada de la consola
                02217   ;       (ver bloque de configuración).
                02218   ;
0339 00         02219   CONIN$Delay$Elapsed:    DB      0       ;Indicador utilizado durante el proceso
                02220                                           ;  de la tecla de función para indicar
                02221                                           ;  el transcurso de un retardo
                02222                                           ;  predeterminado
                02223   ;
                02224   ;===========================
                02225   CONIN:                                  ;<=== Punto de entrada al BDOS (estándar)
                02226   ;===========================
033A 2A8D0F     02227           LHLD    CB$Forced$Input         ;Obtención apuntador a entrada forzada
033D 7E         02228           MOV     A,M                     ;Obtención carácter de entrada siguiente
033E B7         02229           ORA     A                       ;Comprobación de nulo
033F CA4703     02230           JZ      CONIN$No$FI             ;Si, no entrada forzada
0342 23         02231           INX     H                       ;Si, actualizar apuntador
0343 228D0F     02232           SHLD    CB$Forced$Input         ;  y almacenarlo en su posición
0346 C9         02233           RET
                02234   ;
                02235   CONIN$No$FI                             ;No, entrada forzada
0347 2A5800     02236           LHLD    CB$Console$Input        ;Obtención palabra de redireccionamiento
034A 116400     02237           LXI     D,CB$Device$Table$Addresses
034D CD6F06     02238           CALL    Select$Device$Table     ;Obtención dirección tabla de dispositivo
0350 CD9106     02239           CALL    Get$Input$Character     ;Obtención carác. siguiente dispositivo ent.
                02240                                   ;Proceso de teclas-función
                02241
0353 FE1B       02242           CPI     Function$Key$Lead       ;Comprobación de primer carácter de secuencia
                02243                                           ;  de tecla-función (normalmente escape)
0355 C0         02244           RNZ                             ;Si no, retorno a la rutina que llamó al BIOS
0356 F5         02245           PUSH    PSW                     ;Almacenar primer carácter
```

Figura 8-10. (Continuación.)

```
0357 211D00    02246            LXI     H,DT$Function$Delay     ;Obtención constante de tiempo
               02247                                            ;  para esperar la llegada
               02248                                            ;  de caracteres siguientes
               02249                                            ;  de tecla-función
035A 19        02250            DAD     D
035B 4E        02251            MOV     C,M                     ;Obtención valor de retraso
035C 0600      02252            MVI     B,0                     ;Composición en palabra
035E AF        02253            XRA     A                       ;Indicación al temporizador no fuera de tiempo
035F 323903    02254            STA     CONIN$Delay$Elapsed
0362 217B03    02255            LXI     H,CONIN$Set$Delay$Elapsed ;Direc. de reanudación después de la espera
0365 CD6D08    02256            CALL    Set$Watchdog            ;Activación del retardo basada en un reloj de
               02257                                            ;  tiempo real tal que transferirá el control
               02258                                            ;  a una dirección especificada después que
               02259                                            ;  haya transcurrido un intervalo de tiempo
               02260    CONIN$Wait$for$Delay:                   ;Espera hasta que transcurra el retardo
0368 3A3903    02261            LDA     CONIN$Delay$Elapsed     ;Comprobación de indicador activado
036B B7        02262            ORA     A                       ;  por rutina centinela
036C CA6803    02263            JZ      CONIN$Wait$for$Delay
               02264
               02265    CONIN$Check$for$Function:
036F 211900    02266            LXI     H,DT$Character$Count    ;Comprobación de si han entrado el resto de
               02267                                            ;  caracteres de la secuencia
0372 19        02268            DAD     D
0373 7E        02269            MOV     A,M                     ;Obtención cuenta de carac. en la mem. int.
0374 FE02      02270            CPI     Function$Key$Length - 1
0376 D28103    02271            JNC     CONIN$Check$Function    ;Suficientes caracteres en la mem. int.
               02272                                            ;  para posible secuencia de tecla-función
0379 F1        02273            POP     PSW                     ;Insuficientes carac. en la mem. int. para ser
               02274                                            ;  una secuencia de tecla-función, retorno a
               02275                                            ;  la rutina de llamada con el primer carácter
037A C9        02276            RET
               02277
               02278    ;
               02279    ;       La siguiente rutina será llamada por la rutina centinela cuando
               02280    ;       ha transcurrido el retardo especificado
               02281    ;
               02282    CONIN$Set$Delay$Elapsed:
037B 3EFF      02283            MVI     A,0FFH                  ;Indicación temporizador centinela tiempo
037D 323903    02284            STA     CONIN$Delay$Elapsed     ;  excedido
0380 C9        02285            RET                             ;Retorno a rutina centinela
               02286    ;
               02287    ;
               02288    CONIN$Check$Function:
0381 211700    02289            LXI     H,DT$Get$Offset         ;Almacenar "apuntador-obtención" en curso
0384 19        02290            DAD     D                       ;  en la memoria intermedia
0385 7E        02291            MOV     A,M                     ;Obtención apuntador
0386 F5        02292            PUSH    PSW                     ;Almacenar apuntador en la pila
               02293
0387 211700    02294            LXI     H,DT$Get$Offset         ;Comprobación segundo carácter (y posiblemente
038A CDF007    02295            CALL    Get$Address$in$Buffer   ;  tercero) en la secuencia
038D 46        02296            MOV     B,M                     ;Obtención segundo carácter
               02297
               02298            IF      Three$Character$Function
038E C5        02299            PUSH    B                       ;Almacenar para uso posterior
038F 211700    02300            LXI     H,DT$Get$Offset         ;Recuperación tercer carácter
0392 CDF007    02301            CALL    Get$Address$in$Buffer
0395 C1        02302            POP     B                       ;Recuperación segundo carácter
0396 4E        02303            MOV     C,M                     ;Ahora BC = carácter 2, carácter 3
               02304            ENDIF
               02305
0397 D5        02306            PUSH    D                       ;Almacenar apuntador a tabla de dispositivo
0398 21B000    02307            LXI     H,CB$Function$Key$Table - CB$Function$Key$Entry$Size
               02308                                            ;Obtención apuntador tabla de teclas-función
               02309                                            ;  en el bloque de configuración
039B 111300    02310            LXI     D,CB$Function$Key$Entry$Size ;Obten. tamaño ent. preparada para bucle
               02311    CONIN$Next$Function:
039E 19        02312            DAD     D                       ;Desplazar a siguiente (o primera) entrada
039F 7E        02313            MOV     A,M                     ;Obtención segundo carácter de la secuencia
03A0 B7        02314            ORA     A                       ;Comprobación de final tabla de teclas-función
03A1 CAC203    02315            JZ      CONIN$Not$Function      ;Si -- no es una tecla-función
03A4 B8        02316            CMP     B                       ;Comparar segundos caracteres
03A5 C29E03    02317            JNZ     CONIN$Next$Function     ;No coincidencia, intentar ent. sig. en tabla
               02318
               02319            IF      Three$Character$Function
03A8 23        02320            INX     H                       ;HL -> tercer carácter
03A9 7E        02321            MOV     A,M                     ;Obtención tercer carácter de la secuencia
03AA 2B        02322            DCX     H                       ;Simplif. lóg. para secuencias 2 y 3 carac.
```

Figura 8-10. (Continuación.)

```
03AB B9          02323            CMP     C                       ;Comparación terceros caracteres
03AC C29E03      02324            JNZ     CONIN$Next$Function     ;No coincidencia, intentar ent. sig. de la tabla
03AF 23          02325            INX     H                       ;En caso de coincidencia, compensar por
                 02326                                            ;  decrementos extra
                 02327            ENDIF
                 02328
03B0 23          02329            INX     H                       ;HL -> primer carácter de cadena de caracteres
                 02330                                            ;  de sustitución (terminada en byte 00H)
03B1 228D0F      02331            SHLD    CB$Forced$Input         ;La rutina CONIN se encargará de inyectar
                 02332                                            ;  la cadena de sustitución en
                 02333                                            ;  el flujo de entrada
                 02334
                 02335                                            ;Cuando se ha identificado una secuencia
                 02336                                            ;  función, la pila debe ajustarse
                 02337                                            ;  antes de devolver el control
03B4 D1          02338            POP     D                       ;Obtención apuntador tabla dispositivos
03B5 F1          02339            POP     PSW                     ;Sacar de la pila valor desplaz. para "obtención"
03B6 F1          02340            POP     PSW                     ;Sacar de la pila primer carác. de sec. función
                 02341
03B7 211900      02342            LXI     H,DT$Character$Count    ;Decrementar contador de caracteres para tener
03BA 19          02343            DAD     D                       ;  en cuenta los caracteres eliminados de
                 02344                                            ;  la memoria intermedia
03BB 7E          02345            MOV     A,M                     ;Obtención de la cuenta
03BC D602        02346            SUI     Function$Key$Length-1   ; (el primer carácter ya ha sido
03BE 77          02347            MOV     M,A                     ;  deducido)
03BF C33A03      02348            JMP     CONIN                   ;Retorno a CONIN para obtener los
                 02349                                            ;  caracteres de entrada forzada
                 02350    CONIN$Not$Function:
                 02351                                    ;Los intentos de reconocer una secuencia de
                 02352                                    ;  tecla-función han sido infructuosos. El apuntador
                 02353                                    ;  del desplazamiento para "obtención" debe reponerse a
                 02354                                    ;  su antiguo valor para que no se pierda el carácter
                 02355                                    ;  que se supuso formaba parte de la secuencia-función.
                 02356
03C2 D1          02357            POP     D                       ;Recuperación apuntador a tabla de dispositivo
03C3 F1          02358            POP     PSW                     ;Recuperación "desplaz." para "obtención" previo
03C4 211700      02359            LXI     H,DT$Get$Offset
03C7 19          02360            DAD     D                       ;HL -> desplaz. para "obtención" en la tabla
03C8 77          02361            MOV     M,A                     ;Vuelta a valor inicial despl. para "obtención"
                 02362                                            ;  después de detectar el primer carácter
03C9 F1          02363            POP     PSW                     ;Recuperación primer carácter
03CA C9          02364            RET                             ;Devolución del primer carácter al usuario
                 02365    ;
                 02500    ;#
                 02501    ;       Salida de consola
                 02502    ;
                 02503    ;       Esta rutina lanza caracteres de datos al dispositivo de consola.
                 02504    ;       "Atrapa" también secuencias de escape lanzadas a la consola y
                 02505    ;       desencadena acciones específicas de acuerdo con las secuencias.
                 02506    ;       Un autómata-primitivo se utiliza para reconocimiento de secuencias de
                 02507    ;       escape.
                 02508    ;       Además, para lanzar el carácter siguiente a todos los dispositivos
                 02509    ;       seleccionados en la palabra de redireccionamiento de salida de consola,
                 02510    ;       comprueba que no haya sido suspendida la salida del dispositivo
                 02511    ;       seleccionado por un protocolo XON/XOFF y que DTR está alto si debe
                 02512    ;       estarlo.
                 02513    ;       Una vez que se ha lanzado el carácter, si se utiliza el protocolo
                 02514    ;       ETX/ACK y ha salido la longitud especificada del mensaje, se saca un
                 02515    ;       carácter ETX y el dispositivo se marca como suspendido.
                 02516    ;
                 02517    ;       Parámetros de entrada
                 02518    ;
                 02519    ;               C = carácter a lanzar a salida
                 02520    ;
                 02521    ;       Variables de almacenamiento de CONOUT
                 02522    ;
03CB 00          02523    CONOUT$Character:       DB      0       ;Guardar área para carácter a lanzar a salida
                 02524
03CC DB03        02525    CONOUT$Processor:       DW      CONOUT$Normal
                 02526                                            ;Es la dirección del trozo de programa
                 02527                                            ;  que procesará el carácter siguiente.
                 02528                                            ;  El caso por defecto es
                 02529                                            ;  CONOUT$Normal
03CE 0000        02530    CONOUT$String$Pointer:  DW      0       ;Apunta a una cadena (normalmente
                 02531                                            ;  en el bloque de configuración) que es
                 02532                                            ;  prefijado por los caracteres del
                 02533                                            ;  flujo de salida de consola
```

Figura 8-10. (Continuación.)

```
03D0 00         02534   CONOUT$String$Length:   DB      0       ;Contiene el número máximo de caracteres
                02535                                           ;  a prefijar en un flujo de salida
                02536                                           ;  desde consola
                02537
                02538   ;
                02539   ;       *** ATENCION ***
                02540   ;       La rutina de mensajes de error comparte programa en esta subrutina.
                02541   ;       En entrada, el byte de datos a lanzar a la salida estará en la pila
                02542   ;       y el registro DE fijado correctamente.
                02543   ;
                02544   ;
                02545   CONOUT$OEM$Entry:
03D1 32CB03     02546           STA     CONOUT$Character        ;Almacenar byte de datos
03D4 C3E803     02547           JMP     CONOUT$Entry2           ;HL tiene un mapa de bits especial
                02548   ;
                02549   ;=====================
                02550   CONOUT:                         ;<=== Punto de entrada al BIOS (estándar)
                02551   ;=====================
03D7 2ACC03     02552           LHLD    CONOUT$Processor        ;Obtención de la dirección del procesador para
                02553                                           ;  manejo del siguiente carácter a lanzar
                02554                                           ;(El defecto es CONOUT$Normal)
03DA E9         02555           PCHL                            ;Transferencia de control al procesador
                02556   ;
                02557   ;
                02558   CONOUT$Normal:                          ;Procesador normal para salida de consola
03DB 79         02559           MOV     A,C                     ;Comprobación de posible comienzo de secuencia
03DC FE1B       02560           CPI     Function$Key$Lead       ;  de escape
03DE CA1204     02561           JZ      CONOUT$Escape$Found     ;Es posible
                02562   CONOUT$Forced:
03E1 79         02563           MOV     A,C                     ;Punto de entrada salida forzada
03E2 32CB03     02564           STA     CONOUT$Character        ;No es secuencia de escape -- almacenar bytes
                02565                                           ;  de datos
03E5 2A5A00     02566           LHLD    CB$Console$Output       ;Obtención palabra de redirec. de consola
                02567   ;
                02568   CONOUT$Entry2:                  ;<=== Punto de entrada de mensajes de error de salida
                02569   ;
03E8 116400     02570           LXI     D,CB$Device$Table$Addresses     ;Direcciones de tablas de dispositivos
03EB D5         02571           PUSH    D                       ;Puesta en pila preparado para bucle
03EC E5         02572           PUSH    H
                02573
                02574   CONOUT$Next$Device:
03ED E1         02575           POP     H                       ;Recuperación de mapa de bits de redirec.
03EE D1         02576           POP     D                       ;Recuperación apuntador direc. tablas de disp.
03EF CD6F06     02577           CALL    Select$Device$Table     ;Obtención tabla de dispositivos en DE
03F2 B7         02578           ORA     A                       ;Comprobación de si ha sido selec. algún disp.
                02579                                           ;  (mapa de bits no totalmente cero)
03F3 CA0D04     02580           JZ      CONOUT$Exit             ;No, salida
03F6 C5         02581           PUSH    B       ;Yes - B..      ;Almacenar mapa de bits de redireccionamiento
03F7 E5         02582           PUSH    H                       ;Almacenar apuntador direc. tabla dispositivos
                02583   CONOUT$Wait:
03F8 CD0F06     02584           CALL    Check$Output$Ready      ;Comprobación de dispositivo no suspendido y
                02585                                           ;  (si es necesario) DTR-alto
03FB CAF803     02586           JZ      CONOUT$Wait             ;No, espera
                02587
03FE F3         02588           DI                              ;Cancelar interrupciones para prevenir
                02589                                           ;  reentrada involuntaria
03FF 3ACB03     02590           LDA     CONOUT$Character        ;Recuperación byte de datos
0402 4F         02591           MOV     C,A                     ;Preparado para salida
0403 CD2608     02592           CALL    Output$Data$Byte        ;Salida del byte de datos
0406 FB         02593           EI
                02594
0407 CD3A06     02595           CALL    Process$Etx$Protocol    ;Diálogo con protocolo Etx/Ack
040A C3ED03     02596           JMP     CONOUT$Next$Device      ;Bucle para dispositivo siguiente
                02597
                02598   CONOUT$Exit:
040D 3ACB03     02599           LDA     CONOUT$Character        ;Recuperar carácter de datos
0410 79         02600           MOV     A,C                     ;Convenio CP/M
0411 C9         02601           RET
                02602   ;
                02603   CONOUT$Escape$Found:                    ;Posible secuencia de escape
0412 211904     02604           LXI     H,CONOUT$Process$Escape ;Proceso del vector de carácter siguiente
                02605   CONOUT$Set$Processor:
0415 22CC03     02606           SHLD    CONOUT$Processor        ;Fijar dirección vector
0418 C9         02607           RET                             ;Retorno a la rutina que llamó a BIOS
                02700   ;#
                02701   ;
                02702   ;       Salida de consola: Proceso de secuencias de escape
```

Figura 8-10. (Continuación.)

```
                02703   ;
                02704   CONOUT$Process$Escape:                  ;El control llega a este punto con un
                02705                                           ;  carácter después del escape en C
0419 211B02     02706           LXI     H,CONOUT$Escape$Table   ;Obtención base de la tabla de reconocimiento
                02707   CONOUT$Next$Entry:
041C 7E         02708           MOV     A,M                     ;Comprobación de final de tabla
041D B7         02709           ORA     A
041E CA2B04     02710           JZ      CONOUT$No$Match         ;Sí, no coincidencia
0421 B9         02711           CMP     C                       ;Comparación con carácter dato
0422 CA3B04     02712           JZ      CONOUT$Match            ;Coincidencia
0425 23         02713           INX     H                       ;Desplazar a siguiente entrada de tabla
0426 23         02714           INX     H
0427 23         02715           INX     H
0428 C31C04     02716           JMP     CONOUT$Next$Entry       ;Vuelta atrás y nueva comprobación
                02717   ;
                02718   CONOUT$No$Match:                        ;No coincidencia, en consecuencia deben
                02719                                           ;  sacarse escape original
                02720                                           ;  y carácter siguiente
042B C5         02721           PUSH    B                       ;Almacenar carácter después de escape
042C 0E1B       02722           MVI     C,Function$Key$Lead     ;Obtención carácter de escape
042E CDE103     02723           CALL    CONOUT$Forced           ;Salida a dispositivo de consola
0431 C1         02724           POP     B                       ;Obtención de carácter después del escape
0432 CDE103     02725           CALL    CONOUT$Forced           ;Sacarlo también
                02726   ;
                02727   CONOUT$Set$Normal:
0435 21DB03     02728           LXI     H,CONOUT$Normal         ;Fijar vector a valor original para
0438 C31504     02729           JMP     CONOUT$Set$Processor    ;  caracteres siguientes
                02730   ;
                02731
                02732   CONOUT$Match:
043B 23         02733           INX     H                       ;HL -> byte LS de subprocesador
043C 5E         02734           MOV     E,M                     ;Obtención byte LS
043D 23         02735           INX     H
043E 56         02736           MOV     D,M                     ;Obtención byte MS
043F EB         02737           XCHG                            ;HL -> subprocesador
0440 E9         02738           PCHL                            ;Ir a subprocesador
                02739   ;
                02740   CONOUT$Date:                    ;Subprocesador para inyectar fecha en curso
                02741                                   ;  en el flujo de entrada de consola
                02742                                   ;  (utilizando entrada forzada)
0441 218F0F     02743           LXI     H,Date
                02744   CONOUT$Set$Forced$Input:
0444 228D0F     02745           SHLD    CB$Forced$Input
0447 C9         02746           RET                     ;Retorno a la rutina que efectuó la llamada al BIOS
                02747   ;
                02748   CONOUT$Time:                    ;Subprocesador para inyectar la hora en el flujo
                02749                                   ;  de entrada de consola
0448 21990F     02750           LXI     H,Time$In$ASCII
044B C34404     02751           JMP     CONOUT$Set$Forced$Input
                02752   ;
                02753   CONOUT$Set$Date:                ;Subprocesador para fijar la fecha tomando los
                02754                                   ;  8 caracteres siguientes de salida de consola
                02755                                   ;  y almacenarlos en cadena-fecha
044E 21A30F     02756           LXI     H,Time$Date$Flags       ;Fijar indicador para indicar que la fecha
0451 3E02       02757           MVI     A,Date$Set              ;  ha sido fijada por programa
0453 B6         02758           ORA     M
0454 77         02759           MOV     M,A
0455 3E08       02760           MVI     A,8                     ;Fijar cuenta de caracteres
0457 218F0F     02761           LXI     H,Date                  ;Fijar dirección
045A C36C04     02762           JMP     CONOUT$Set$String$Pointer
                02763   ;
                02764   ;
                02765   CONOUT$Set$Time:                ;Subprocesador para fijar la hora tomando los
                02766                                   ;  8 caracteres siguientes de salida de consola
                02767                                   ;  y almacenarlos en cadena-hora
045D 21A30F     02768           LXI     H,Time$Date$Flags       ;Fijar indicador para indicar que la hora
0460 3E01       02769           MVI     A,Time$Set              ;  ha sido fijada por programa
0462 B6         02770           ORA     M
0463 77         02771           MOV     M,A
0464 3E08       02772           MVI     A,8                     ;Fijar cuenta de caracteres
0466 21990F     02773           LXI     H,Time$in$ASCII         ;Fijar dirección
0469 C36C04     02774           JMP     CONOUT$Set$String$Pointer
                02775   ;
                02776   CONOUT$Set$String$Pointer:              ;HL -> cadena, A = contador
046C 32D003     02777           STA     CONOUT$String$Length    ;Almacenar cuenta
046F 22CE03     02778           SHLD    CONOUT$String$Pointer   ;Almacenar dirección
0472 217804     02779           LXI     H,CONOUT$Process$String ;Vector salida adicional
```

Figura 8-10. (Continuación.)

```
0475 C31504     02780           JMP     CONOUT$Set$Processor
                02781   ;
                02782   CONOUT$Process$String:          ;El control llega a este punto para
                02783                                   ;  cada carácter en la cadena en el registro C.
                02784                                   ;  El carácter se apila en la cadena
                02785                                   ;  de recepción hasta que se encuentra un byte 00
                02786                                   ;  o hasta que se alcanza el número de
                02787                                   ;  caracteres especificado.
0478 2ACE03     02788           LHLD    CONOUT$String$Pointer   ;Obtención de la direc. para pila de carac.
047B 79         02789           MOV     A,C                     ;Comprobación de carácter en curso 00H
047C B7         02790           ORA     A
047D CA3504     02791           JZ      CONOUT$Set$Normal       ;Vuelta al proceso normal
0480 77         02792           MOV     M,A                     ;En caso contrario, carácter en la pila
0481 23         02793           INX     H                       ;Actualizar apuntador
0482 3600       02794           MVI     M,00H                   ;Poner en la fila terminal
0484 22CE03     02795           SHLD    CONOUT$String$Pointer   ;Almacenar apuntador actualizado
0487 21D003     02796           LXI     H,CONOUT$String$Length  ;Decrementar contador
048A 35         02797           DCR     M
048B CA3504     02798           JZ      CONOUT$Set$Normal       ;Vuelta al proceso normal
                02799                                           ;  si el contador llega a 0
048E C9         02800           RET                             ;Retorno a
                02801                                           ;  CONOUT$Process$Strings
                02802   ;
                02900   ;#
                02901   ;
                02902   ;       Estado de entrada auxiliar
                02903   ;
                02904   ;       Esta rutina comprueba la cuenta de caracteres en la memoria
                02905   ;       intermedia de entrada adecuada.
                02906   ;       El registro A toma un valor indicando si existen o no datos
                02907   ;       a la espera.
                02908   ;
                02909   ;       Parámetros de entrada: ninguno.
                02910   ;
                02911   ;       Parámetros de salida
                02912   ;
                02913   ;               A = 000H si no hay datos esperando
                02914   ;               A = 0FFH si los hay
                02915   ;
                02916   ;=========================
                02917   AUXIST:                         ;<=== Punto de entrada al BIOS (privado)
                02918   ;=========================
048F 2A5C00     02919           LHLD    CB$Auxiliary$Input              ;Obtención palabra de redirec.
0492 116400     02920           LXI     D,CB$Device$Table$Addresses     ;  y apuntador de tabla
0495 CD6F06     02921           CALL    Select$Device$Table     ;Obtención de dirección tabla de dispositivo
0498 C34708     02922           JMP     Get$Input$Status        ;Obtención del estado de disp. de entrada
                02923                                           ;  y retorno a quien efectuó la llamada
                02924   ;
                03000   ;#
                03001   ;
                03002   ;       Estado de salida auxiliar
                03003   ;
                03004   ;       Esta rutina fija el valor del registro A para indicar cuándo el
                03005   ;       dispositivo(s) auxiliar está preparado(s) para aceptar datos de salida.
                03006   ;       Puede utilizarse más de un dispositivo para salida
                03007   ;       auxiliar. Esta rutina devuelve el AND booleano de sus estados.
                03008   ;
                03009   ;       Parámetros de entrada: ninguno
                03010   ;
                03011   ;       Parámetros de salida
                03012   ;
                03013   ;               A = 000H si no están preparados uno o más dispositivos
                03014   ;               A = 0FFH si todos los dispositivos listado están preparados
                03015   ;
                03016   ;
                03017   ;======================
                03018   AUXOST:                         ;<=== Punto de entrada al BIOS (privado)
                03019   ;======================
049B 2A5E00     03020           LHLD    CB$Auxiliary$Output             ;Obtención palabra de redirec. de list.
049E C37905     03021           JMP     Get$Composite$Status
                03022   ;
                03100   ;#
                03101   ;
                03102   ;       Entrada auxiliar (sustitución para lectora)
                03103   ;
                03104   ;       Esta rutina devuelve el siguiente carácter de entrada
```

Figura 8-10. (Continuación.)

```
                03105   ;       del dispositivo lógico auxiliar apropiado.
                03106   ;
                03107   ;       Parámetros de entrada: ninguno.
                03108   ;
                03109   ;       Parámetros de salida
                03110   ;
                03111   ;               A = carácter-dato
                03112   ;
                03113   ;=========================
                03114   AUXIN:                          ;<=== Punto de entrada al BIOS (estándar)
                03115   ;=========================
04A1 2A5C00     03116           LHLD    CB$Auxiliary$Input          ;Obtención palabra de redirec.
04A4 116400     03117           LXI     D,CB$Device$Table$Addresses ;  y apuntador a tabla
04A7 CD6F06     03118           CALL    Select$Device$Table     ;Obtención dirección tabla de dispositivo
04AA C39106     03119           JMP     Get$Input$Character     ;Obtención carácter de entrada siguiente
                03120                                           ;  y retorno a quien efectuó la llamada
                03121   ;
                03200   ;#
                03201   ;       Salida auxiliar (sustituye PUNCH)
                03202   ;
                03203   ;       Esta rutina lanza un byte de datos al dispositivo(s) auxiliar.
                03204   ;       Es similar a CONOUT excepto en que utiliza el
                03205   ;       temporizador centinela para detectar si un dispositivo
                03206   ;       permanece ocupado más de 30 segundos consecutivos.
                03207   ;       Si ello ocurre, lanza un mensaje a la consola.
                03208   ;
                03209   ;       Parámetros de entrada
                03210   ;
                03211   ;               C = byte de datos
                03212   ;
04AD 0D0A07417503213   AUXOUT$Busy$Message:    DB      CR,LF,7,'Auxiliary device not Ready?',CR,LF,0
                03214   ;
                03215   ;=====================
                03216   AUXOUT:                         ;<=== Punto de entrada al BIOS (estándar)
                03217   ;=====================
04CE 2A5E00     03218           LHLD    CB$Auxiliary$Output ;Obtención palabra de redirec. auxiliar
04D1 11AD04     03219           LXI     D,AUXOUT$Busy$Message ;Mensaje a lanzar a salida si se
                03220                                         ;  sobrepasa límite de tiempo
04D4 C3A205     03221           JMP     Multiple$Output$Byte
                03222   ;
                03300   ;#
                03301   ;
                03302   ;       Estado de impresión
                03303   ;
                03304   ;       Esta rutina fija el registro A para indicar cuándo el
                03305   ;       dispositivo(s) de impresión está preparado(s) para aceptar datos.
                03306   ;       Puede utilizarse más de un dispositivo de listado de salida.
                03307   ;       La rutina devuelve el AND booleano de todos los estados.
                03308   ;
                03309   ;       Parámetros de entrada: ninguno.
                03310   ;
                03311   ;       Parámetros de salida
                03312   ;
                03313   ;               A = 000H si al menos uno de los dispositivos no está preparado
                03314   ;               A = 0FFH si todos los dispositivos están preparados
                03315   ;
                03316   ;
                03317   ;=====================
                03318   LISTST:                                 ;<=== Punto de entrada al BIOS (estándar)
                03319   ;=====================
04D7 2A6200     03320           LHLD    CB$List$Output          ;Obtención palabra de redirec. de listado
04DA C37905     03321           JMP     Get$Composite$Status
                03322   ;
                03400   ;#
                03401   ;       Salida de impresión
                03402   ;
                03403   ;       Esta rutina lanza al dispositivo de listado un byte de datos.
                03404   ;       Es similar a CONOUT excepto en que utiliza el
                03405   ;       temporizador centinela para detectar si la impresora
                03406   ;       está ocupada más de 30 segundos consecutivos.
                03407   ;       Si ello ocurre, lanza un mensaje a la consola.
                03408   ;
                03409   ;       Parámetros de entrada
                03410   ;
                03411   ;               C = byte de datos
                03412   ;
```

Figura 8-10. (Continuación.)

```
04DD 0D0A07507203413   LIST$Busy$Message:      DB      CR,LF,7,'Printer not Ready?',CR,LF,0
                03414   ;
                03415   ;=====================
                03416   LIST:                           ;<=== Punto de entrada al BIOS (estándar)
                03417   ;=====================
04F5 2A6200     03418           LHLD    CB$List$Output          ;Obtención palabra de redirec. de listados
04F8 11DD04     03419           LXI     D,LIST$Busy$Message     ;Mensaje a lanzar si se sobrepasa
                03420                                           ;  el tiempo límite
04FB C3A205     03421           JMP     Multiple$Output$Byte
                03422   ;
                03500   ;#
                03501   ;       Petición de elección al usuario
                03502   ;
                03503   ;       Esta rutina visualiza un mensaje de error pidiendo la
                03504   ;       elección de:
                03505   ;
                03506   ;               R -- Intentar de nuevo la operación que ha causado el error
                03507   ;               I -- Ignorar el error e intentar continuación
                03508   ;               A -- Abortar el programa y volver a CP/M
                03509   ;
                03510   ;       Esta rutina acepta un carácter de la consola, lo convierte
                03511   ;       a mayúsculas y devuelve control a quien hizo
                03512   ;       la llamada con la respuesta en el registro A.
                03513   ;
                03514   RUC$Message:
04FE 0D0A       03515                   DB      CR,LF
0500 2020202020 03516                   DB      '       Enter R - Retry, I - Ignore, A - Abort : ',
                03517   ;
                03518   ;
                03519   Request$User$Choice:
052F CD2D03     03520           CALL    CONST                   ;Engullir cualquier mecanografiado-en-avance
0532 CA3B05     03521           JZ      RUC$Buffer$Empty
0535 CD3A03     03522           CALL    CONIN
0538 C32F05     03523           JMP     Request$User$Choice
                03524
                03525   RUC$Buffer$Empty:
053B 21FE04     03526           LXI     H,RUC$Message           ;Visualizar indicación
053E CD5305     03527           CALL    Output$Error$Message
                03528
0541 CD3A03     03529           CALL    CONIN                   ;Obtención de carácter de consola
0544 CD3B0E     03530           CALL    A$To$Upper              ;Poner en mayúsculas para comparación
0547 32B00D     03531           STA     Disk$Action$Confirm     ;Almacenar en mensaje de confirmación
054A F5         03532           PUSH    PSW                     ;Almacenar para uso posterior
                03533
054B 21B00D     03534           LXI     H,Disk$Action$Confirm
054E CD5305     03535           CALL    Output$Error$Message
                03536
0551 F1         03537           POP     PSW                     ;Recuperar código de acción
0552 C9         03538           RET
                03539   ;
                03600   ;#
                03601   ;
                03602   ;       Salida de mensaje de error
                03603   ;
                03604   ;       Esta rutina lanza un mensaje de error a todos los dispositivos de consola
                03605   ;       seleccionados en curso excepto a aquellos que están siendo utilizados
                03606   ;       para recibir salida de impresión. Esto se hace para prevenir
                03607   ;       situaciones de "abrazos mortales" en los que el hecho de que la
                03608   ;       impresora esté ocupada durante demasiado tiempo causa la salida
                03609   ;       de un mensaje de error -- y la salida se dirige a la propia impresora.
                03610   ;
                03611   ;       Esta subrutina hace uso de buena parte de la rutina CONOUT.
                03612   ;       Para economizar memoria, entra en CONOUT utilizando un punto
                03613   ;       de entrada privado.
                03614   ;
                03615   ;       Parámetros de entrada
                03616   ;
                03617   ;       HL -> mensaje de error terminado en byte 00
                03618   ;
                03619   Output$Error$Message:
0553 E5         03620           PUSH    H                       ;Guardar dirección mensaje
0554 2A5A00     03621           LHLD    CB$Console$Output       ;Obtención mapa de bits de redic. de consola
0557 EB         03622           XCHG
0558 2A6200     03623           LHLD    CB$List$Output          ;Obtención mapa de bits de redirec. de listado
                03624                                           ;HL = impresora, DE = consola
                03625                                           ;Fijar a 0 todos los bits en el mapa de bits
```

Figura 8-10. (Continuación.)

```
                    03626                                   ;  de la consola que estén a uno en el mapa
                    03627                                   ;  de bits de la impresora
055B 7C             03628           MOV     A,H             ;Obtener byte MS de impresora
055C 2F             03629           CMA                     ;Inversión
055D A2             03630           ANA     D               ;Preservar sólo los bits a cero
055E 67             03631           MOV     H,A             ;Salvar resultado
055F 7D             03632           MOV     A,L             ;Repetir para byte LS de la impresora
0560 2F             03633           CMA
0561 A3             03634           ANA     E
0562 6F             03635           MOV     L,A             ;HL contiene ahora únicamente dispositivos
                    03636                                   ;  de consola puros
0563 B4             03637           ORA     H               ;Asegurar que al menos un dispositivo
0564 CA6A05         03638           JZ      OEM$Device$Present  ;  esté seleccionado
0567 210100         03639           LXI     H,0001H         ;En caso contrario, utilizar disp. por defec.0
                    03640   OEM$Device$Present:
                    03641   OEM$Next$Character:
056A D1             03642           POP     D               ;Recuperar dirección de mensaje en DE
056B 1A             03643           LDAX    D               ;Obtención byte siguiente del mensaje
056C 13             03644           INX     D               ;Actualización apuntador a mensaje
056D B7             03645           ORA     A               ;Comprobación de final de mensaje
056E C8             03646           RZ                      ;Si, salida
056F D5             03647           PUSH    D               ;Salvar direc. de mensaje para uso posterior
0570 E5             03648           PUSH    H               ;Almacenar mapa de bits especial
                    03649                                   ;Carácter-dato en A
0571 CDD103         03650           CALL    CONOUT$OEM$Entry    ;Entrar programa compartido
0574 E1             03651           POP     H               ;Recuperación mapa de bits especial
0575 C36A05         03652           JMP     OEM$Next$Character
                    03653   ;
                    03654   ;
                    03655   ;
                    03656   ;       Obtención de estado compuesto
                    03657   ;
                    03658   ;       Esta rutina fija el registro A para indicar cuándo el dispositivo(s)
                    03659   ;       de salida está preparado para aceptar datos de salida.
                    03660   ;       Puede utilizar más de un dispositivo de salida. Esta
                    03661   ;       rutina devuelve el AND booleano de los estados.
                    03662   ;
                    03663   ;       Parámetros de entrada
                    03664   ;
                    03665   ;               HL = mapa de bits de redirec. de E/S para dispositivos de salida
                    03666   ;
                    03667   ;       Parámetros de salida
                    03668   ;
                    03669   ;               A = 000H si uno de los dispositivos no está preparado
                    03670   ;               A = 0FFH si están todos preparados
                    03671   ;
0578 00             03672   GCS$Status:     DB      0       ;Estado compuesto de todos los dispositivos
                    03673   ;
                    03674   ;
                    03675   Get$Composite$Status:
0579 3EFF           03676           MVI     A,0FFH          ;Suposición todos los dispositivos preparados
057B 327805         03677           STA     GCS$Status      ;Prefijar byte de estado compuesto
                    03678
057E 116400         03679           LXI     D,CB$Device$Table$Addresses    ;Direc. de tablas de dispositivos
0581 D5             03680           PUSH    D               ;Puesta en la pila preparada para bucle
0582 E5             03681           PUSH    H               ;Almacenar mapa de bits
                    03682   GCS$Next$Device:
0583 E1             03683           POP     H               ;Recuperación mapa de bits de redirec.
0584 D1             03684           POP     D               ;Recuperación apuntador a direc. tabla disp.
0585 CD6F06         03685           CALL    Select$Device$Table ;Obtención tabla de dispositivos en DE
0588 B7             03686           ORA     A               ;Comprobación de si un disp. ha sido selec.
                    03687                                   ;  (mapa de bits no idénticamente nulo)
0589 CA9905         03688           JZ      GCS$Exit        ;No, salida
058C C5             03689           PUSH    B       ;Yes - B..  ;Almacenar mapa de bits de redireccionamiento
058D E5             03690           PUSH    H               ;Almacenar apuntador a direc. tabla de disp.
058E CD0F06         03691           CALL    Check$Output$Ready  ;Comprobación de dispositivo preparado
0591 217805         03692           LXI     H,GCS$Status    ;AND con los estados de los
0594 A6             03693           ANA     M               ;  dispositivos previos
0595 77             03694           MOV     M,A             ;Almacenar estado compuesto
                    03695
0596 C38305         03696           JMP     GCS$Next$Device     ;Bucle para dispositivo siguiente
                    03697   ;
                    03698   GCS$Exit:
0599 3A7805         03699           LDA     GCS$Status      ;Retorno con estado compuesto
059C B7             03700           ORA     A
059D C9             03701           RET
```

Figura 8-10. (Continuación.)

```
                03702   ;
                03800   ;#
                03801   ;
                03802   ;       Byte de salida múltiple
                03803   ;
                03804   ;       Esta rutina lanza a salida un byte de datos a todos los dispositivos
                03805   ;       especificados en la palabra de redireccionamiento de entrada/salida.
                03806   ;       Es similar a CONOUT excepto en que utiliza el temporizador centinela
                03807   ;       para detectar si alguno de los dispositivos
                03808   ;       permanece ocupado más de 30 segundos consecutivos.
                03809   ;       Si ello ocurre, lanza un mensaje a la consola.
                03810   ;
                03811   ;       Parámetros de entrada
                03812   ;
                03813   ;               HL = mapa de bits de redireccionamiento E/S
                03814   ;               DE -> mensaje a lanzar si se sobrepasa el límite de tiempo
                03815   ;               C = byte de datos
                03816   ;
0708 =          03817   MOB$Maximum$Busy       EQU     1800    ;Número de pulsos de reloj (uno cada
                03818                                           ;  16,66 ms) durante los que el dispositivo
                03819                                           ;  puede estar ocupado
059E 00         03820   MOB$Character:          DB      0       ;Carácter a lanzar a salida
059F 0000       03821   MOB$Busy$Message:       DW      0       ;Dirección del mensaje a lanzar si se
                03822                                           ;  sobrepasa el límite de tiempo
05A1 00         03823   MOB$Need$Message:       DB      0       ;Indicador utilizado para detectar que se ha
                03824                                           ;  sobrepasado límite temporizador centinela
                03825   ;
                03826   Multiple$Output$Byte:
05A2 79         03827           MOV     A,C                     ;Obtención byte de datos
05A3 329E05     03828           STA     MOB$Characters          ;Almacenar copia
05A6 EB         03829           XCHG                            ;HL -> mensaje de límite de tiempo
05A7 229F05     03830           SHLD    MOB$Busy$Message        ;Salvar para uso posterior
05AA EB         03831           XCHG                            ;Mapa de bits
                03832
05AB 116400     03833           LXI     D,CB$Device$Table$Addresses     ;Direc. de tablas de dispositivos
05AE D5         03834           PUSH    D                       ;Almacenar en fila preparada para bucle
05AF E5         03835           PUSH    H                       ;Almacenar mapa de bits de redirec. de E/S
                03836   MOB$Next$Device:
05B0 E1         03837           POP     H                       ;Recuperación mapa de bits de redirec.
05B1 D1         03838           POP     D                       ;Recuperac. apuntador direc. de tabla de disp.
05B2 CD6F06     03839           CALL    Select$Device$Table     ;Obtención tabla de dispositivos en DE
05B5 B7         03840           ORA     A                       ;Comprobación de existencia de disp. selec.
05B6 CAEC05     03841           JZ      MOB$Exit
                03842
05B9 C5         03843           PUSH    B       ;<- Yes : B     ;Almacenar apuntador a direc. tabla disp.
05BA E5         03844           PUSH    H                       ;Almacenar mapa de bits de redireccionamiento
                03845   ;
                03846   MOB$Start$Watchdog:
05BB AF         03847           XRA     A                               ;Puesta a cero del indicador de
05BC 32A105     03848           STA     MOB$Need$Message                ;  mensaje que se precisa
05BF 010807     03849           LXI     B,MOB$Maximum$Busy              ;Retardo de tiempo
05C2 210906     03850           LXI     H,MOB$Not$Ready                 ;Dirección a dirigirse
05C5 CD6D08     03851           CALL    Set$Watchdog                    ;Arranque de temporizador
                03852
                03853   MOB$Wait:
05C8 3AA105     03854           LDA     MOB$Need$Message                ;Comprobación de tiempo centinela exc.
05CB B7         03855           ORA     A
05CC C2EE05     03856           JNZ     MOB$Output$Message              ;Sí, mensaje de atención
05CF CD0F06     03857           CALL    Check$Output$Ready              ;Comprobación dispositivo preparado
05D2 CAC805     03858           JZ      MOB$Wait                        ;No, espera
                03859   ;
05D5 F3         03860           DI                                      ;Desactivar interrup. para prevenir
                03861                                                   ;  reentrada involuntaria
05D6 010000     03862           LXI     B,0                             ;Desconectar temporizador centinela
05D9 CD6D08     03863           CALL    Set$Watchdog                    ; (valor fijado en HL irrelevante)
                03864
05DC 3A9E05     03865           LDA     MOB$Character                   ;Obtención de byte de datos
05DF 4F         03866           MOV     C,A
05E0 CD2608     03867           CALL    Output$Data$Byte                ;Salida de byte de datos
05E3 FB         03868           EI
05E4 CD3A06     03869           CALL    Process$Etx$Protocol            ;Diálogo con protocolo ETX/ACK
05E7 C3B005     03870           JMP     MOB$Next$Device
                03871   ;
                03872   MOB$Ignore$Exit:                                ;Ignorar error de tiempo excedido
05EA E1         03873           POP     H                               ;Ajustar la pila
05EB D1         03874           POP     D
```

Figura 8-10. (Continuación.)

```
                03875   ;
                03876   MOB$Exit:
05EC 79         03877           MOV     A,C                     ;"Convenios" CP/M
05ED C9         03878           RET
                03879   ;
                03880   MOB$Output$Message:
05EE 2A9F05     03881           LHLD    MOB$Busy$Message        ;Visualización mensaje de atención en los
05F1 CD5305     03882           CALL    Output$Error$Message    ;  dispositivos de consola seleccionados
                03883   MOB$Request$Choice:
05F4 CD2F05     03884           CALL    Request$User$Choice     ;Visualización de mensaje y obtención de
                03885                                           ;  carácter de acción
05F7 FE52       03886           CPI     'R'                     ;Nuevo intento
05F9 CABB05     03887           JZ      MOB$Start$Watchdog      ;Conectar centinela e intentar de nuevo
05FC FE49       03888           CPI     'I'                     ;Ignorar
05FE CAEA05     03889           JZ      MOB$Ignore$Exit
0601 FE41       03890           CPI     'A'                     ;Aborto
0603 CA360E     03891           JZ      System$Reset            ;  Dar función 0 del BDOS
0606 C3F405     03892           JMP     MOB$Request$Choice
                03893   ;
                03894   MOB$Not$Ready:                  ;La rutina de temporizador-centinela llamará a ésta
                03895                                   ;  si el dispositivo está ocupado más de 30 segundos
                03896                                   ;  aproximadamente
                03897                                   ;NOTA: Esta es una rutina del servicio de interrup.
0609 3EFF       03898           MVI     A,0FFH                  ;Elevar petición de salida de mensaje
060B 32A105     03899           STA     MOB$Need$Message
060E C9         03900           RET                             ;Retorno a rutina centinela
                03901   ;
                04000   ;#
                04001   ;       Comprobación salida preparada
                04002   ;
                04003   ;       Esta rutina comprueba si el dispositivo especificado
                04004   ;       está preparado para recibir datos de salida.
                04005   ;       Para ello debe comprobar si el dispositivo ha sido suspendido
                04006   ;       por razones de protocolo y si DTR está bajo.
                04007   ;
                04008   ;       NOTA: Esta rutina no comprueba si la USART está preparada.
                04009   ;             Este test se realiza en la propia rutina de salida de byte de datos.
                04010   ;
                04011   ;       Parámetros de entrada
                04012   ;
                04013   ;               DE -> tabla de dispositivos
                04014   ;
                04015   ;       Parámetros de salida
                04016   ;
                04017   ;               A = 000H (indicador de cero a uno): Dispositivo no preparado
                04018   ;               A = 0FFH (indicador de cero a cero): Dispositivo preparado
                04019   ;
                04020   Check$Output$Ready:
060F 210E00     04021           LXI     H,DT$Status             ;Obtención estado del dispositivo
0612 19         04022           DAD     D                       ;HL -> byte de estado
0613 7E         04023           MOV     A,M                     ;Obtención byte de estado
0614 47         04024           MOV     B,A                     ;Realizar una copia del byte de estado
0615 E601       04025           ANI     DT$Output$Suspend       ;Comprobación de salida suspendida
0617 C23806     04026           JNZ     COR$Not$Ready           ;Sí, indicar no preparado
                04027
061A 3E04       04028           MVI     A,DT$Output$DTR         ;Comprobar si DTR debe estar alto para mandar
061C A0         04029           ANA     B                       ;Enmascarar con estado del disp. de la tabla
061D CA3406     04030           JZ      COR$Ready               ;No, dispositivo preparado lógicamente
                04031
0620 210000     04032           LXI     H,DT$Status$Port        ;Poner a 1 para leer estado del dispositivo
0623 19         04033           DAD     D
0624 7E         04034           MOV     A,M                     ;Obtención número de puerto estado
0625 322906     04035           STA     COR$Status$Port         ;Puesto a uno instrucción posterior
                04036
0628 DB         04037           DB      IN
                04038   COR$Status$Port:
0629 00         04039           DB      0               ;<-- Puesto a uno por instrucción anterior
062A 4F         04040           MOV     C,A                     ;Almacenar estado del hardware
                04041
062B 210400     04042           LXI     H,DT$DTR$Ready          ;Sí, puesta a uno para comprobación estado del
062E 19         04043           DAD     D                       ;  chip para ver si DTR está alto
062F 7E         04044           MOV     A,M                     ;Obtención máscara de estado de DTR alto
0630 A1         04045           ANA     C                       ;Comprobar estado del chip
0631 CA3806     04046           JZ      COR$Not$Ready           ;DTR bajo indica no preparado
                04047   ;
                04048   COR$Ready:
```

Figura 8-10. (Continuación.)

```
0634 3EFF       04049           MVI     A,0FFH                  ;Indicar dispositivo preparado para salida
0636 B7         04050           ORA     A
0637 C9         04051           RET
                04052   ;
                04053   COR$Not$Ready:                          ;Indicar dispositivo no preparado para salida
0638 AF         04054           XRA     A
0639 C9         04055           RET
                04056   ;
                04200   ;#
                04201   ;
                04202   ;       Procesado del protocolo ETX/ACK
                04203   ;
                04204   ;       Esta rutina soporta el protocolo ETX/ACK.
                04205   ;       Después de que un número especificado de caracteres de datos
                04206   ;       ha sido lanzado al dispositivo, se lanza un carácter ETX y el dispositivo
                04207   ;       se pone en estado suspendido. Se reanuda la salida al dispositivo
                04208   ;       únicamente después de recibir un carácter de entrada ACK (enviado
                04209   ;       bajo control de interrupciones).
                04210   ;
                04211   ;       Parámetros de entrada
                04212   ;
                04213   ;               DE -> tabla de dispositivo
                04214   ;
                04215   ;       Parámetros de salida
                04216   ;
                04217   ;               Cuenta de mensajes decrementada (y puesta a cero si es necesario)
                04218   ;
                04219   Process$Etx$Protocol:
063A 210E00     04220           LXI     H,DT$Status             ;Comprobación disponibilidad protocolo ETX/ACK
063D 19         04221           DAD     D
063E 7E         04222           MOV     A,M
063F E610       04223           ANI     DT$Output$Etx
0641 C8         04224           RZ                              ;No, retorno inmediato
0642 211000     04225           LXI     H,DT$Etx$Count          ;Sí, decrementar contador
0645 19         04226           DAD     D
0646 E5         04227           PUSH    H                       ;Almacenar direc. contador para uso posterior
0647 4E         04228           MOV     C,M                     ;Obtención byte LS
0648 23         04229           INX     H
0649 46         04230           MOV     B,M                     ;Obtención byte MS
064A 0B         04231           DCX     B
064B 78         04232           MOV     A,B
064C B1         04233           ORA     C                       ;Comprobación cuenta igual a cero
064D C25706     04234           JNZ     PEP$Save$Count          ;No
0650 211200     04235           LXI     H,DT$Etx$Message$Length ;Sí, puesta a cero longitud de mensaje
0653 19         04236           DAD     D
0654 4E         04237           MOV     C,M                     ;Obtención byte LS
0655 23         04238           INX     H
0656 46         04239           MOV     B,M                     ;Obtención byte MS
                04240   PEP$Save$Count:
0657 E1         04241           POP     H                       ;Recuperación dirección de contador
0658 71         04242           MOV     M,C                     ;Salvar cuenta en la tabla
0659 23         04243           INX     H
065A 70         04244           MOV     M,B
                04245   ;
065B B7         04246           ORA     A                       ;Restablecer cuando contador alcanza 0
065C C0         04247           RNZ                             ;No, no es necesario más proceso
065D 0E03       04248           MVI     C,ETX                   ;Sí, envío de ETX al dispositivo
065F F3         04249           DI                              ;Previsión de reentrada involuntaria
0660 CD2608     04250           CALL    Output$Data$Byte
0663 FB         04251           EI
0664 210E00     04252           LXI     H,DT$Status             ;Indicador de salida suspendida
0667 19         04253           DAD     D
0668 F3         04254           DI                              ;Previsión de interacción con interrupción
0669 7E         04255           MOV     A,M                     ;Obtención byte de estado
066A F601       04256           ORI     DT$Output$Suspend       ;Fijar bit
066C 77         04257           MOV     M,A                     ;Almacenar en tabla
066D FB         04258           EI
066E C9         04259           RET
                04260   ;
                04400   ;#
                04401   ;
                04402   ;       Selección de tabla de dispositivo
                04403   ;
                04404   ;       Esta rutina examina una palabra de 16 bits y, dependiendo del primer bit
                04405   ;       que encuentra a uno, selecciona la dirección de la tabla de dispositivo que
                04406   ;       le corresponde.
```

Figura 8-10. (Continuación.)

```
                04407   ;       Parámetros de entrada
                04408   ;
                04409   ;               HL = Mapa de bits
                04410   ;               DE -> Tabla de direcciones de tablas de dispositivos
                04411   ;                       La primera dirección de la tabla se selecciona
                04412   ;                       cuando el bit menos significativo del mapa
                04413   ;                       de bits está a uno, y así sucesivamente
                04414   ;
                04415   ;       Parámetros de salida
                04416   ;
                04417   ;               BC -> Entrada en curso en las direcciones de tabla de dispositivos
                04418   ;               DE = Dirección de la tabla de dispositivo seleccionada
                04419   ;               HL = Mapa de bits desplazado
                04420   ;                    Distinto de cero si se ha encontrado un bit a uno
                04421   ;                    Cero si el mapa de bits es idénticamente 0
                04422   ;
                04423   ;       Nota: Si, en entrada, HL es 0000H, será devuelta en DE la primera entrada
                04424   ;       en las direcciones de tabla de dispositivo.
                04425   ;
                04426   Select$Device$Table:
066F 7C         04427           MOV     A,H             ;Obtención byte más significativo del mapa de bits
0670 B5         04428           ORA     L               ;Comprobación de si HL es nulo completamente
0671 C8         04429           RZ                      ;Retorno indicando que no hay más bits a uno
0672 7D         04430           MOV     A,L             ;Comprobación de si byte LS es no nulo
0673 E601       04431           ANI     1
0675 C28006     04432           JNZ     SDT$Bit$Set     ;Sí, devolución de la dirección correspondiente
0678 13         04433           INX     D               ;No, actualizar apuntador a tabla
0679 13         04434           INX     D
067A CDDB08     04435           CALL    SHLR            ;Desplazar HL un bit a la derecha
067D C36F06     04436           JMP     Select$Device$Table     ;Comprobación bit siguiente
                04437   SDT$Bit$Set:
0680 E5         04438           PUSH    H               ;Almacenar mapa de bit desplazado
0681 42         04439           MOV     B,D             ;Tomar copia de tabla de apuntadores
0682 4B         04440           MOV     C,E
0683 EB         04441           XCHG                    ;HL -> dirección en la tabla
0684 5E         04442           MOV     E,M
0685 23         04443           INX     H
0686 56         04444           MOV     D,M             ;DE -> tabla de dispositivo seleccionada
                04445                                   ;Activar registros para una nueva
                04446                                   ;  entrada
0687 E1         04447           POP     H               ;Recuperación mapa de bits desplazado
0688 CDDB08     04448           CALL    SHLR            ;Desplazamiento mapa de bits un lugar a la derecha
068B 03         04449           INX     B               ;Actualización apuntador a tabla de direcciones DT
068C 03         04450           INX     B               ;  a la entrada actual
068D 3E01       04451           MVI     A,1             ;Indicación de que se ha encontrado un bit a uno
068F B7         04452           ORA     A               ;  y registros activados correctamente
0690 C9         04453           RET
                04454   ;
                04600   ;#
                04601   ;
                04602   ;       ;Obtención de carácter de entrada
                04603   ;
                04604   ;       Esta rutina obtiene el siguiente carácter de entrada del
                04605   ;       dispositivo especificado en la tabla de dispositivos
                04606   ;       manejándolo como un parámetro de entrada.
                04607   ;
                04608   Get$Input$Character:
0691 211900     04609           LXI     H,DT$Character$Count    ;Comprobación de si han sido almacenados
0694 19         04610           DAD     D                       ;  caracteres en la memoria intermedia
                04611   GIC$Wait:
0695 FB         04612           EI                              ;Asegurar que serán detectados los caracteres
                04613                                           ;  de entrada que lleguen
0696 7E         04614           MOV     A,M                     ;Obtención cuenta de caracteres
0697 B7         04615           ORA     A
0698 CA9506     04616           JZ      GIC$Wait                ;No hay caracteres, espera
069B 35         04617           DCR     M                       ;Decrementar contador de caracteres habida
                04618                                           ;  cuenta del carácter que se ha extraído
                04619                                           ;  de la memoria intermedia
069C 211700     04620           LXI     H,DT$Get$Offset         ;Utilización del despl. obtención para acceso
069F CDF007     04621           CALL    Get$Address$in$Buffer   ;Retorno de HL -> carácter y con
                04622                                           ;  desplazamiento-para-obtención actualizado
06A2 7E         04623           MOV     A,M                     ;Obtención carácter dato en curso
06A3 F5         04624           PUSH    PSW                     ;Almacenar para utilización posterior
                04625
06A4 211900     04626           LXI     H,DT$Character$Count    ;Comprob. cuenta de carac. decrementada en la
06A7 19         04627           DAD     D                       ;  mem. intr.,comprobando si puede existir ent.
```

Figura 8-10. (Continuación.)

```
                    04628
06A8 7E             04629           MOV     A,M                     ;  reanudarse después de haber sido suspendida
06A9 211B00         04630           LXI     H,DT$Resume$Input$Count ;Obtención cuenta decrementada
06AC 19             04631           DAD     D                       ;Comprobación de si cuenta en curso coincide
06AD BE             04632           CMP     M                       ;  con umbral de memoria intermedia vacía
06AE C2DB06         04633           JNZ     GIC$Check$Control       ;No en el umbral, comprobación de entrada de
                    04634                                           ;  carácter de control
06B1 210E00         04635           LXI     H,DT$Status             ;En el umbral, comprobación de la forma
06B4 19             04636           DAD     D                       ;  utilizada para espaciar la entrada
06B5 F3             04637           DI                              ;Asegurar no interacción de las interrupciones
                    04638                                           ;  durante el cambio del byte de estado
06B6 7E             04639           MOV     A,M                     ;Obtención del byte de estado/protocolo
06B7 E6FD           04640           ANI     OFFH AND NOT DT$Input$Suspend    ;Borrado de indicador de suspensión
06B9 77             04641           MOV     M,A                     ;Almacenamiento del estado actualizado
06BA FB             04642           EI
06BB F5             04643           PUSH    PSW                     ;Guardar para uso posterior
06BC E640           04644           ANI     DT$Input$RTS            ;Comprob. si se ha realiz. el borr. para enviar
06BE CAD006         04645           JZ      GIC$Check$Input$Xon     ;No
06C1 210B00         04646           LXI     H,DT$RTS$Control$Port   ;Sí, obtención número puerto control
06C4 19             04647           DAD     D
06C5 7E             04648           MOV     A,M
06C6 32CF06         04649           STA     GIC$Raise$RTS$Port              ;Almacenar en instrucción posterior
06C9 210D00         04650           LXI     H,DT$Raise$RTS$Value
06CC 19             04651           DAD     D
06CD 7E             04652           MOV     A,M                     ;Obtención del valor necesario para elevar RTS
                    04653
06CE D3             04654           DB      OUT
                    04655   GIC$Raise$RTS$Port:
06CF 00             04656           DB      0               ;Fijación en instrucción posterior
                    04657                                       ;Salto para comprobación de XON
                    04658   GIC$Check$Input$Xon:             ;Comprobación de si ha sido utilizado XON/XOF para
                    04659                                    ;  la suspensión temporal de la entrada
06D0 F1             04660           POP     PSW                     ;Recuperación del byte de ent./prot.
06D1 E680           04661           ANI     DT$Input$Xon            ;Comprobación de si el bit XON está a uno
06D3 CADB06         04662           JZ      GIC$Check$Control       ;No, ver entrada de carácter de control
06D6 0E11           04663           MVI     C,XON                   ;Sí, salida de carácter XON
06D8 CD2608         04664           CALL    Output$Data$Byte
                    04665
                    04666   GIC$Check$Control:
06DB CD0808         04667           CALL    Check$Control$Char      ;Comprobación de si es un carác. de control
06DE CAE606         04668           JZ      GIC$Not$Control         ;No, es CR, LF o TAB
06E1 211C00         04669           LXI     H,DT$Control$Count      ;Sí, decremento de contador de control de
06E4 19             04670           DAD     D                       ;  caracteres en la memoria intermedia
06E5 35             04671           DCR     M
                    04672   GIC$Not$Control:
06E6 F1             04673           POP     PSW                     ;Recuperación carácter dato
06E7 C9             04674           RET
                    04675
                    04676   ;
                    04800   ;*
                    04801   ;
                    04802   ;***************************************************
                    04803   ;*                                                 *
                    04804   ;* Control de interrup. para unidades de ent./sal. serie *
                    04805   ;*                                                 *
                    04806   ;***************************************************
                    04807   ;
                    04808   ;       Se han diseñado estas unidades para capturar caracteres de entrada
                    04809   ;       utilizando interrupciones generadas por hardware. La salida no está
                    04810   ;       controlada por interrupción, ya que, en general, el programa usuario
                    04811   ;       puede generar la salida bastante más rápidamente que el hardware puede
                    04812   ;       sacarla; por tanto, la salida se pondrá rápidamente en un "estado
                    04813   ;       estacionario" esperando ser tomada en cuenta por el hardware.
                    04814   ;
                    04815   ;
                    04816   ;       Las unidades de entrada soportan las siguientes características:
                    04817   ;
                    04818   ;               o   Protocolos serie para control de salida
                    04819   ;                       DTR alto para enviar
                    04820   ;                       XON/XOF
                    04821   ;                       ETX/ACK con longitud de mensaje variable
                    04822   ;               o   Protocolos serie para control de entrada
                    04823   ;                       RTS alto para enviar
                    04824   ;                       XON/XOF
                    04825   ;
                    04826   ;       Los caracteres de entrada se almacenan en una memoria intermedia
```

Figura 8-10. (Continuación.)

```
                04827   ;       circular hasta que el programa usuario está preparado para aceptarlos.
                04828   ;       Opcionalmente, las unidades pueden prever el borrado de esta memoria
                04829   ;       intermedia utilizando RTS bajo o XOF para pedir la interrupción de la
                04830   ;       entrada.
                04831   ;
                04832   ;
                04833   ;       Rutina de interrupción de carácter.
                04834   ;
                04835   ;       Esta rutina recibe el control cuando el hardware detecta un carácter
                04836   ;       de entrada. Los chips del hardware y las posiciones de memoria
                04837   ;       necesarias se habrán fijado durante el programa de inicialización.
                04838   ;
                04839   ;
                04840   ;       Como consecuencia de que realmente las arquitecturas varían
                04841   ;       enormemente, esta rutina ha sido escrita para mostrar genéricamente el
                04842   ;       proceso requerido. Supone una estructura de interrupciones que
                04843   ;       transfiere el control a esta rutina sea cual sea el dispositivo
                04844   ;       realmente interrumpido. Supone también que una vez que ha entrado un
                04845   ;       carácter, se genera un valor específico de salida a un puerto particular
                04846   ;       para preparar de nuevo el hardware para una nueva interrupción.
                04900   ;#
                04901   ;
                04902   Character$Interrupt:
06E8 E5         04903           PUSH    H               ;Salvar HL en pila de usuario
06E9 210000     04904           LXI     H,0             ;Obtención del valor en curso del apuntador a pila
06EC 39         04905           DAD     SP
06ED 228422     04906           SHLD    PI$User$Stack   ;Salvar pila de usuario
06F0 31B022     04907           LXI     SP,PI$Stack     ;Cambio a pila local
06F3 F5         04908           PUSH    PSW             ;Salvar resto de registros
06F4 C5         04909           PUSH    B
06F5 D5         04910           PUSH    D
                04911
                04912                           ;Comprobación de cada dispositivo por orden para detección
                04913                           ;  de caracteres de entrada
06F6 11BE02     04914           LXI     D,DT$0          ;Dispositivo 0
06F9 CD1607     04915           CALL    Service$Device  ;Si interrumpido entrada del carácter
                04916                                   ;  y proceso de acuerdo con él y el estado
                04917                                   ;  del dispositivo
06FC 11AE02     04913           LXI     D,DT$1          ;Dispositivo 1

06FF CD1607     04919           CALL    Service$Device
                04920
0702 11CE02     04921           LXI     D,DT$2          ;Dispositivo 2
0705 CD1607     04922           CALL    Service$Device
                04923
0708 3E20       04924           MVI     A,IC$EOI        ;Indicación a chip controlador de interrupciones
070A D3D8       04925           OUT     IC$OCW2$Port    ;  de que ha sido atendida la interrupción
070C D1         04926           POP     D               ;Restaurar valor de registros
070D C1         04927           POP     B
070E F1         04928           POP     PSW
070F 2A8422     04929           LHLD    PI$User$Stack   ;Conmutar a pila de usuario
0712 F9         04930           SPHL
0713 E1         04931           POP     H
0714 FB         04932           EI                      ;Activar de nuevo interrupciones en la CPU
0715 C9         04933           RET                     ;Reanudar proceso interrumpido
                04934   ;
                05000   ;#
                05001   ;
                05002   ;       Servicio de dispositivos
                05003   ;
                05004   ;       Esta rutina configura el servicio de interrupciones de dispositivo,
                05005   ;       comprobando si el dispositivo especificado en la tabla de dispositivo
                05006   ;       (dirección en DE) está interrumpido actualmente, en este caso entra el
                05007   ;       carácter. Dependiendo del carácter entrado, esta rutina puede almacenarlo en
                05008   ;       la memoria intermedia de entrada (haciendo "caer" el flujo de entrada si la
                05009   ;       memoria intermedia está prácticamente llena) o puede suspender o reanudar
                05010   ;       la salida al dispositivo.
                05011   ;
                05012   ;       Parámetros de entrada
                05013   ;
                05014   ;               DE -> tabla de dispositivo
                05015   ;
                05016   Service$Device:
0716 210000     05017           LXI     H,DT$Status$Port        ;Comprobación de si este dispositivo
0719 19         05018           DAD     D                       ;  está realmente interrumpido
```

Figura 8-10. (Continuación.)

```
071A 7E        05019          MOV     A,M                     ;Obtención número de puerto de estado
071B 321F07    05020          STA     SD$Status$Port          ;Almacenar en instrucción posterior
               05021
071E DB        05022          DB      IN                      ;Entrada de estado
               05023  SD$Status$Port:
071F 00        05024          DB      0       ;<-- Fijar según instrucción anterior
               05025  ;
0720 210300    05026          LXI     H,DT$Input$Ready        ;Comp. de si el estado indica prep. para datos
0723 19        05027          DAD     D
0724 A6        05028          ANA     M                       ;Enmascarar con valor de entrada preparada
0725 C8        05029          RZ                              ;No, retorno al servicio de interrupción
               05030                                          ;Comprobación de ocurrencia de errores
0726 210700    05031          LXI     H,DT$Detect$Error$Port  ;Activar interrupción para leer estado
0729 19        05032          DAD     D                       ;  de error
072A 7E        05033          MOV     A,M                     ;Obtención número de puerto de estado
072B 322F07    05034          STA     SD$Error$Port           ;Almacenar en instrucción posterior
               05035
072E DB        05036          DB      IN                      ;Entrada de estado de error
               05037  SD$Error$Port:
072F 00        05038          DB      0       ;<-- Fijado por instrucción anterior
               05039  ;
0730 210800    05040          LXI     H,DT$Detect$Error$Value ;Mask with error bit(s)
0733 19        05041          DAD     D
0734 A6        05042          ANA     M
0735 CA4707    05043          JZ      SD$No$Error             ;No bit(s) a uno
0738 210900    05044          LXI     H,DT$Reset$Error$Port   ;Activar para puesta a cero del error
073B 19        05045          DAD     D
073C 7E        05046          MOV     A,M                     ;Obtención número de puerto de puesta a cero
073D 324607    05047          STA     SD$Reset$Error$Port     ;Almacenar en instrucción posterior
0740 210A00    05048          LXI     H,DT$Reset$Error$Value
0743 19        05049          DAD     D
0744 7E        05050          MOV     A,M                     ;Obtención valor de puesta a cero de interrup.
               05051
0745 D3        05052          DB      OUT
               05053  SD$Reset$Error$Port:
0746 00        05054          DB      0       ;<-- Fijada por instrucción anterior
               05055
               05056  SD$No$Error:
0747 210100    05057          LXI     H,DT$Data$Port          ;Entrada de caracteres-dato (puede ser
074A 19        05058          DAD     D                       ;  "mutilado" si ocurre un error)
074B 7E        05059          MOV     A,M                     ;Obtención número de puerto de datos
074C 325007    05060          STA     SD$Data$Port            ;Almacenar en instrucción posterior
               05061
074F DB        05062          DB      IN                      ;Entrada de carácter-dato
               05063  SD$Data$Port:
0750 00        05064          DB      0       ;<-- Fijada por instrucción anterior
               05065
0751 47        05066          MOV     B,A                     ;Hacer copia carácter de dato anterior
0752 210E00    05067          LXI     H,DT$Status             ;Comprobación de protocolo activo en curso
0755 19        05068          DAD     D                       ;  XON o ETX
0756 7E        05069          MOV     A,M                     ;Obtención byte de protocolo
0757 E618      05070          ANI     DT$Output$Xon + DT$Output$Etx
0759 CA8107    05071          JZ      SD$No$Protocol          ;Ninguno está activo
075C E608      05072          ANI     DT$Output$Xon           ;Comprobación de si XON/XOFF activo
075E C26E07    05073          JNZ     SD$Check$if$Xon         ;Sí, comprobación de entrada de carácter XON
               05074                                          ;No, suponer que ETX/ACK está activo
0761 3E06      05075          MVI     A,ACK                   ;Comprobación por si carácter de entrada ACK
0763 B8        05076          CMP     B
0764 C28107    05077          JNZ     SD$No$Protocol          ;No, procesar carácter como dato
               05078  SD$Output$Desuspend:                    ;Sí, dispositivo preparado para aceptar
               05079                                          ;  más datos, indicar al dispositivo
               05080                                          ;  la reanudación
               05081                                          ;La rutina de sal. de control con no interrup.
               05082                                          ;  comprueba el bit de suspensión
0767 7E        05083          MOV     A,M                     ;Obtención de byte de estado/protocolo
0768 E6FE      05084          ANI     0FFH AND NOT DT$Output$Suspend ;Preser. todos los bits excep. el susp.
076A 77        05085          MOV     M,A                     ;Salvar con bit de suspensión = 0
076B C3D907    05086          JMP     SD$Exit                 ;Salida al servicio de interrupciones sin
               05087                                          ;  almacenar carácter dato
               05088  ;
               05089  SD$Check$if$Xon:                        ;Protocolo XON/XOFF activo, de forma que
               05090                                          ;  si se recibe XOFF suspensión de la salida
               05091                                          ;  si se recibe XON reanudarla
               05092                                          ;La rutina de sal. de control con no interrup.
               05093                                          ;  comprueba el bit de suspensión
076E 3E11      05094          MVI     A,XON                   ;Comprobación de entrada de carácter XON
```

Figura 8-10. (Continuación.)

```
0770 B8          05095          CMP     B
0771 CA6707      05096          JZ      SD$Output$Desuspend      ;Sí, preparación salida al dispositivo
0774 3E13        05097          MVI     A,XOFF                   ;Comprobación de entrada de carácter XOFF
0776 B8          05098          CMP     B
0777 C28107      05099          JNZ     SD$No$Protocol           ;No, procesar carácter como dato
                 05100  SD$Output$Suspend:                      ;El dispositivo precisa una pausa en la sal.
                 05101                                          ;  de datos para indicar suspensión de salida
077A 7E          05102          MOV     A,M                      ;Obtención de byte de estado/protocolo
077B F601        05103          ORI     DT$Output$Suspend        ;Fijar a uno bit de suspensión
077D 77          05104          MOV     M,A                      ;Almacenar en tabla de dispositivo
077E C3D907      05105          JMP     SD$Exit                  ;Salida al servicio de interrupciones sin
                 05106                                          ;  almacenar carácter de entrada
                 05107  ;
                 05108  SD$No$Protocol:
0781 211800      05109          LXI     H,DT$Buffer$Length$Mask  ;Comprobación de si queda espacio en la
0784 19          05110          DAD     D                        ;  memoria intermedia de entrada
0785 7E          05111          MOV     A,M                      ;Obtención longitud - 1
0786 3C          05112          INR     A                        ;Actualizar longitud verdadera
0787 211900      05113          LXI     H,DT$Character$Count     ;Obtención cuenta en curso de caracteres
078A 19          05114          DAD     D                        ;  en la memoria intermedia
078B BE          05115          CMP     M                        ;Comprobación longitud = cuenta
078C CAEB07      05116          JZ      SD$Buffer$Full           ;Sí, salida de carácter asociado a sonido
078F C5          05117          PUSH    B                        ;Almacenar carácter nulo
0790 211600      05118          LXI     H,DT$Put$Offset          ;Cálculo de la dirección del carácter en la
                 05119                                          ;  memoria intermedia de entrada
0793 CDF007      05120          CALL    Get$Address$In$Buffer    ;HL -> posición del carácter
0796 C1          05121          POP     B                        ;Recuperación carácter de entrada
0797 70          05122          MOV     M,B                      ;Salvar carácter en la mem. int. de entrada
                 05123                                  ;Actualizar número de caracteres en la
                 05124                                  ;  memoria intermedia, comprobando si la
                 05125                                  ;  entrada debe detenerse temporalmente
0798 211900      05126          LXI     H,DT$Character$Count
079B 19          05127          DAD     D
079C 34          05128          INR     M                        ;Actualización de la cuenta de caracteres
079D 7E          05129          MOV     A,M                      ;Obtención contador actualizado
079E 211A00      05130          LXI     H,DT$Stop$Input$Count    ;Comprobación de si cuenta en curso coincide
07A1 19          05131          DAD     D                        ;  con el umbral de memoria intermedia
07A2 BE          05132          CMP     M                        ;  completa
07A3 C2CE07      05133          JNZ     SD$Check$Control         ;No en el umbral, comprobación de
                 05134                                          ;  entrada de carácter de control
07A6 210E00      05135          LXI     H,DT$Status              ;En el umbral, comprobar en qué forma debe
07A9 19          05136          DAD     D                        ;  detenerse momentáneamente la entrada
07AA 7E          05137          MOV     A,M                      ;Obtención byte de estado/ protocolo
07AB F602        05138          ORI     DT$Input$Suspend         ;Indicación de entrada suspendida
07AD 77          05139          MOV     M,A                      ;Almacenar estado actualizado en tabla
07AE F5          05140          PUSH    PSW                      ;Almacenamiento para utilización posterior
07AF E640        05141          ANI     DT$Input$RTS             ;Comprob. de si debe ignorarse bajo-para-salida
07B1 CAC307      05142          JZ      SD$Check$Input$Xon       ;No
07B4 210B00      05143          LXI     H,DT$RTS$Control$Port    ;Sí, obtención número de puerto de control
07B7 19          05144          DAD     D
07B8 7E          05145          MOV     A,M
07B9 32C207      05146          STA     SD$Drop$RTS$Port         ;Almacenamiento en instrucción posterior
07BC 210C00      05147          LXI     H,DT$Drop$RTS$Value
07BF 19          05148          DAD     D
07C0 7E          05149          MOV     A,M                      ;Obtención de valor necesario para salto RTS
                 05150
07C1 D3          05151          DB      OUT
                 05152  SD$Drop$RTS$Port:
07C2 00          05153          DB      0                ;<- Fijada en instrucción anterior
                 05154                                           ;Salto a test de entrada XON
                 05155  SD$Check$Input$Xon:               ;Comprobación de si se está utilizando protocolo
                 05156                                   ;  XON/XOFF para suspender temporalmente la entrada
07C3 F1          05157          POP     PSW                      ;Recuperación de estado/protocolo
07C4 E680        05158          ANI     DT$Input$Xon             ;Comprobación de bit XON a uno
07C6 CACE07      05159          JZ      SD$Check$Control         ;No, ver control de entrada de caracteres
07C9 0E13        05160          MVI     C,XOFF                   ;Sí, salida de un carácter XOFF
07CB CD2608      05161          CALL    Output$Data$Byte         ;Salida de byte de datos
                 05162  ;
                 05163  SD$Check$Control:                ;Comprobación de si existe en entrada un carácter de
                 05164                                   ;  control (distinto de CR, LF o TAB) y actualizar
                 05165                                   ;  cuenta de caracteres de control en la mem. int.
07CE CD0808      05166          CALL    Check$Control$Char       ;Comprobación de caracteres de control
07D1 CAD907      05167          JZ      SD$Exit                  ;No, no existe ningún carácter de control
07D4 211C00      05168          LXI     H,DT$Control$Count
07D7 19          05169          DAD     D
07D8 34          05170          INR     M                        ;Actualización cuenta de caracteres
```

Figura 8-10. (Continuación.)

```
                        05171   ;
                        05172   SD$Exit:                        ;Puesta a cero del sistema de interrupciones hardware
07D9 210500             05173           LXI     H,DT$Reset$Int$Port
07DC 19                 05174           DAD     D
07DD 7E                 05175           MOV     A,M                     ;Obtención de núm. o de puerto de puesta a cero
07DE B7                 05176           ORA     A                       ;Comprobación de puerto especificado
                        05177                                           ;  (suposición de que será NZ)
07DF C8                 05178           RZ                              ;Ignorar puesta a cero si puerto no especif.
07E0 32E907             05179           STA     SD$Reset$Int$Port       ;Almacenamiento en instrucción posterior
07E3 210600             05180           LXI     H,DT$Reset$Int$Value
07E6 19                 05181           DAD     D
07E7 7E                 05182           MOV     A,M                     ;Obt. de valor de puesta a cero de interrup.
                        05183
07E8 D3                 05184           DB      OUT
                        05185   SD$Reset$Int$Port:
07E9 00                 05186           DB      0       ;<-- Fijada por instrucción anterior
07EA C9                 05187           RET                             ;Retorno a la rutina de servicio de interrup.
                        05188   ;
                        05189   SD$Buffer$Full:                         ;Mem. int. de entrada completamente llena
07EB 0E07               05190           MVI     C,BELL                  ;Envío de carácter sónico como última
07ED C32608             05191           JMP     Output$Data$Byte        ;  medida. Nótese que el retorno JMP se hará
                        05192                                           ;  por subrutina
                        05193   ;
                        05300   ;#
                        05301   ;
                        05302   ;       Obtención de dirección en memoria intermedia
                        05303   ;
                        05304   ;       Esta rutina calcula la dirección del siguiente carácter a acceder
                        05305   ;       a la memoria intermedia.
                        05306   ;
                        05307   ;       Parámetros de entrada
                        05308   ;
                        05309   ;               DE -> tabla de dispositivo apropiada
                        05310   ;               HL =  desplazamiento en tabla de dispositivo o bien
                        05311   ;                       Get$Offset o bien Put$Offset
                        05312   ;
                        05313   ;       Parámetros de salida
                        05314   ;
                        05315   ;               DE no cambia
                        05316   ;               HL -> dirección en memoria intermedia de carácter
                        05317   ;
                        05318   Get$Address$In$Buffer:
07F0 19                 05319           DAD     D                       ;HL -> obtención/puesta desplaz. en tabla disp.
07F1 E5                 05320           PUSH    H                       ;Preservar apuntador a tabla
07F2 4E                 05321           MOV     C,M                     ;Obtención valor del desplazamiento
07F3 0600               05322           MVI     B,0                     ;Composición en valor palabra
                        05323                                   ;Actualización valor del desplazamiento, poniéndolo a
                        05324                                   ;  cero al final de la memoria intermedia
07F5 79                 05325           MOV     A,C                     ;Obtención copia del desplazamiento
07F6 3C                 05326           INR     A                       ;Actualización a nueva posición
07F7 211800             05327           LXI     H,DT$Buffer$Length$Mask
07FA 19                 05328           DAD     D
07FB A6                 05329           ANA     M                       ;Enmascarar bits LS con longitud - 1
07FC E1                 05330           POP     H                       ;Recuperación apuntador a desplaz. en tabla
07FD 77                 05331           MOV     M,A                     ;Almac. nuevo valor (fijado a 0 si es nec.)
07FE 211400             05332           LXI     H,DT$Buffer$Base        ;Obtención direc. base de la mem. int. de entr.
0801 19                 05333           DAD     D                       ;HL -> direc. de la mem. int. en la tabla
0802 7E                 05334           MOV     A,M                     ;Obtención byte LS de dirección
0803 23                 05335           INX     H                       ;HL -> byte MS de dirección
0804 66                 05336           MOV     H,M                     ;H = byte MS
0805 6F                 05337           MOV     L,A                     ;L = byte LS
0806 09                 05338           DAD     B                       ;Añadir base a desplazamiento
0807 C9                 05339           RET
                        05340
                        05341   ;
                        05400   ;#
                        05401   ;
                        05402   ;       Comprobación de control de caracteres
                        05403   ;
                        05404   ;       Esta rutina comprueba el carácter en A para ver si se trata
                        05405   ;       de un carácter de control distinto de CR, LF o TAB. El resultado
                        05406   ;       se devuelve en el indicador Z.
                        05407   ;
                        05408   ;       Parámetros de entrada
                        05409   ;
                        05410   ;               A = carácter a comprobar
```

Figura 8-10. (Continuación.)

```
                        05411   ;
                        05412   ;       Parámetros de salida
                        05413   ;
                        05414   ;               Estado cero si A no contiene un carácter de control
                        05415   ;                       distinto de CR, LF o TAB.
                        05416   ;
                        05417   ;               Distinto de cero si A contiene un carácter de control
                        05418   ;                       distinto de CR, LF o TAB.
                        05419   Check$Control$Char:
0808 3E1F               05420           MVI     A,' ' - 1               ;El espacio es el primer carácter no de
080A B8                 05421           CMP     B                       ;  control
080B DA2408             05422           JC      CCC$No                  ;No se trata de un carácter de control
080E 3E0D               05423           MVI     A,CR                    ;Comprobación de si es de retorno de carro
0810 B8                 05424           CMP     B
0811 CA2408             05425           JZ      CCC$No                  ;No es un carácter de control
0814 3E0A               05426           MVI     A,LF                    ;Comprobación de si es de LF
0816 B8                 05427           CMP     B
0817 CA2408             05428           JZ      CCC$No                  ;No es un carácter de control
081A 3E09               05429           MVI     A,TAB                   ;Comprobación de si es tab. horizontal
081C B8                 05430           CMP     B
081D CA2408             05431           JZ      CCC$No                  ;No es un carácter de control
0820 3E01               05432           MVI     A,1                     ;Indicación de un carácter de control
0822 B7                 05433           ORA     A
0823 C9                 05434           RET
                        05435   CCC$No:                                 ;Indicación de que A no contiene
0824 AF                 05436           XRA     A                       ;  un carácter de control
0825 C9                 05437           RET
                        05438   ;
                        05500   ;#
                        05501   ;
                        05502   ;       Salida de byte de datos
                        05503   ;
                        05504   ;       Esta es una rutina simple de salida que lanza un simple carácter
                        05505   ;       (en el registro C en entrada) al dispositivo especificado en la tabla
                        05506   ;       de dispositivos.
                        05507   ;       Por preferencia, esta rutina habría sido reentrante. Sin embargo,
                        05508   ;       debe almacenar los números de puerto. Por tanto, para utilizarla
                        05509   ;       con un programa ejecutándose con las interrupciones activadas, la secuencia
                        05510   ;       de instrucciones debe ser:
                        05511   ;
                        05512   ;               DI              Interrupciones inhibidas
                        05513   ;               CALL    Output$Data$Byte
                        05514   ;               EI              ;Activa la interrupción
                        05515   ;
                        05516   ;       Dejar de hacer esto puede ser causa de una reentrada involuntaria
                        05517   ;
                        05518   ;       Parámetros de entrada
                        05519   ;
                        05520   ;               C = carácter para salida
                        05521   ;               DE -> tabla de dispositivos
                        05522   ;
                        05523   Output$Data$Byte:
0826 C5                 05524           PUSH    B                       ;Salvar registros
0827 210200             05525           LXI     H,DT$Output$Ready       ;Obtención de máscara de estado de salida
082A 19                 05526           DAD     D                       ;  preparada
082B 46                 05527           MOV     B,M
082C 210000             05528           LXI     H,DT$Status$Port        ;Obtención de número de puerto de estado
082F 19                 05529           DAD     D
0830 7E                 05530           MOV     A,M
0831 323508             05531           STA     ODB$Status$Port         ;Almacenamiento en instrucción posterior
                        05532   ODB$Wait$until$Ready:
                        05533
0834 DB                 05534           DB      IN                      ;Lectura de estado
                        05535   ODB$Status$Port:
0835 00                 05536           DB      0       ;<-- Fijado en instrucción anterior
                        05537
0836 A0                 05538           ANA     B                       ;Comprobación de si está preparado para salida
0837 CA3408             05539           JZ      ODB$Wait$until$Ready    ;No
083A 210100             05540           LXI     H,DT$Data$Port          ;Obtención de puerto de datos
083D 19                 05541           DAD     D
083E 7E                 05542           MOV     A,M
083F 324408             05543           STA     ODB$Data$Port           ;Almacenamiento en instrucción posterior
0842 79                 05544           MOV     A,C                     ;Obtención carácter para salida
                        05545
0843 D3                 05546           DB      OUT
                        05547   ODB$Data$Port:
```

Figura 8-10. (Continuación.)

```
0844 00          05548           DB      0       ;<-- Fijado en instrucción anterior
                 05549
0845 C1          05550           POP     B                       ;Realmacenamiento registros
0846 C9          05551           RET
                 05552   ;
                 05700   ;#
                 05701   ;
                 05702   ;
                 05703   ;       Rutina de estado de entrada
                 05704   ;
                 05705   ;       Esta rutina devuelve un valor al registro A que indica si uno o más caracteres
                 05706   ;       de datos está esperando en la memoria intermedia de entrada. Algunos productos,
                 05707   ;       tales como el Microsoft BASIC, impiden la continuación del mecanografiado
                 05708   ;       normal, engullendo continuamente caracteres, con el fin de comprobar si se
                 05709   ;       ha recibido un Control-S, -Q o -C de entrada. Con el fin de preservar lo
                 05710   ;       mecanografiado en-avance en estas circunstancias, el retorno de estado de
                 05711   ;       entrada puede, como opción escogida por el usuario, devolver los "datos que
                 05712   ;       esperan" sólo si la memoria intermedia de entrada contiene un Control-S, -Q o -C.
                 05713   ;       Esto permite la continuación del mecanografiado engañando al Microsoft BASIC.
                 05714   ;
                 05715   ;       Parámetros de entrada
                 05716   ;
                 05717   ;               DE -> tabla de dispositivos
                 05718   ;
                 05719   ;       Parámetros de salida
                 05720   ;
                 05721   ;               A = 000H si no hay ningún carácter esperando en la
                 05722   ;                       memoria intermedia de entrada
                 05723   ;
                 05724   ;
                 05725   Get$Input$Status:
0847 210F00      05726           LXI     H,DT$Status$2           ;Comprob. de si está disp. en modo simulado
084A 19          05727           DAD     D                       ;HL -> byte de estado en tabla
084B 7E          05728           MOV     A,M                     ;Obtención del byte de estado
084C E601        05729           ANI     DT$Fake$Typeahead       ;Aislamiento del bit de estado
084E CA5B08      05730           JZ      GIS$True$Status         ;Modo de simulación desconectado
                 05731                                   ;
                 05732                                   ;Modo de simulación -- indicando sólo datos preparados
                 05733                                   ;si hay caracteres de control en la mem. int.
0851 211C00      05734           LXI     H,DT$Control$Count      ;Comprobación de si hay caracteres de control
0854 19          05735           DAD     D                       ;  en la memoria intermedia de entrada
0855 AF          05736           XRA     A                       ;Cheap 0
0856 B6          05737           ORA     M                       ;Fijación de indicador según cuenta
0857 C8          05738           RZ                              ;Retorno indicando cero
                 05739   GIS$Data$Ready:
0858 AF          05740           XRA     A                       ;Cheap 0
0859 3D          05741           DCR     A                       ;Fijación de A = 0FFH e indicador NZ
085A C9          05742           RET                             ;Retorno al llamador
                 05743   ;
                 05744   GIS$True$Status:                ;
                 05745                                   ;Estado verdadero, basado en algunos caracteres
                 05746                                   ;preparados en la memoria intermedia de entrada
085B 2A8D0F      05747           LHLD    CB$Forced$Input         ;Comprob. si hay alguna ent. forzada esperando
085E 7E          05748           MOV     A,M                     ;Obtención del próximo carác. de ent. forzada
085F B7          05749           ORA     A                       ;Comprobación de si es distinto a cero
0860 C25808      05750           JNZ     GIS$Data$Ready          ;Sí, indica que hay datos esperando
                 05751
0863 211900      05752           LXI     H,DT$Character$Count    ;Comprobación de si hay caracteres
0866 19          05753           DAD     D                       ;  en la memoria intermedia
0867 7E          05754           MOV     A,M                     ;Obtención de cuenta de caracteres
0868 B7          05755           ORA     A
0869 C8          05756           RZ                              ;Memoria intermedia vacía, A = 0, Z-set
086A C35808      05757           JMP     GIS$Data$Ready
                 05758   ;
                 05759   ;
                 05900   ;#
                 05901   ;
                 05902   ;       Proceso de reloj de tiempo real
                 05903   ;
                 05904   ;       El control se transfiere a la rutina RTC$Interrupt RTC en cada pulso
                 05905   ;       del reloj de tiempo real. Se va decrementando la cuenta de pulsos
                 05906   ;       para comprobar si ha transcurrido un segundo completo. Si es así,
                 05907   ;       se actualiza el tiempo ASCII del bloque de configuración.
                 05908   ;
                 05909   ;       Con cada pulso, la cuenta de reloj-centinela se decrementa para ver si
                 05910   ;       el control debe ser "forzado" a una dirección especificada previamente
```

Figura 8-10. (Continuación.)

```
                05911   ;       al regreso de la interrupción RTC. El temporizador-centinela
                05912   ;       puede utilizarse para transferir el control fuera de lo que puede ser
                05913   ;       un bucle infinito, como es el de esperar que la impresora esté preparada.
                05914   ;
                05915   ;
                05916   ;       Fijación de centinela
                05917   ;
                05918   ;       Esta es una subrutina de nivel de no interrupción que simplemente fija
                05919   ;       la cuenta y la dirección del centinela.
                05920   ;
                05921   ;       Parámetros de entrada
                05922   ;
                05923   ;               BC = número de pulsos de reloj antes de que el centinela esté
                05924   ;                    "fuera de tiempo"
                05925   ;               HL = dirección a la cual se transferirá el control cuando
                05926   ;                    el centinela esté fuera de tiempo
                05927   ;
                05928   Set$Watchdog:
086D F3         05929           DI                              ;Evita interferencia de las interrupciones
086E 22C100     05930           SHLD    RTC$Watchdog$Address    ;Fijación de dirección
0871 60         05931           MOV     H,B
0872 69         05932           MOV     L,C
0873 22BF00     05933           SHLD    RTC$Watchdog$Count      ;Fijación de cuenta
0876 FB         05934           EI
0877 C9         05935           RET
                05936   ;
                05937   ;
                06000   ;#
                06001   ;
                06002                           ;El control se recibe aquí en cada pulso de
                06003                           ;  reloj de tiempo real
                06004   RTC$Interrupt:
0878 F5         06005           PUSH    PSW                     ;Almacenamiento de otros registros
0879 228622     06006           SHLD    PI$User$HL              ;Conmutación a la pila local
087C 210000     06007           LXI     H,0
087F 39         06008           DAD     SP                      ;Obtención de pila de usuarios
0880 228422     06009           SHLD    PI$User$Stack           ;Almacenamiento
0883 31B022     06010           LXI     SP,PI$Stack             ;Conmutación a pila local
0886 C5         06011           PUSH    B
0887 D5         06012           PUSH    D
                06013
0888 21BE00     06014           LXI     H,RTC$Tick$Count        ;Decrementar la cuenta de pulsos
088B 35         06015           DCR     M
088C C2B008     06016           JNZ     RTC$Check$Watchdog      ;No está aún a 0
                06017                                           ;Ha transcurrido un segundo por consiguiente
088F 3ABD00     06018           LDA     RTC$Ticks$per$Second    ;  puesta al valor original
0892 77         06019           MOV     M,A
                06020                                           ;Actualización del reloj de tiempo real ASCII
0893 11A10F     06021           LXI     D,Time$in$ASCII$End     ;DE -> 1 carácter después de hora ASCII
0896 21BD00     06022           LXI     H,Update$Time$End       ;HL -> 1 carácter después de tabla de control
                06023   RTC$Update$Digit:
0899 1B         06024           DCX     D                       ;Decrementar apuntador a hora ASCII
089A 2B         06025           DCX     H                       ;Decrementar apuntador a tabla de control
089B 7E         06026           MOV     A,M                     ;Obtención del próximo carácter de control
089C B7         06027           ORA     A                       ;Comprobación de final de tabla y por consig.
089D CAB008     06028           JZ      RTC$Clock$Updated       ;  todos los díg. del reloj están actualizados
08A0 FA9908     06029           JM      RTC$Update$Digit        ;Saltar ":" en la hora ASCII
08A3 1A         06030           LDAX    D                       ;Obtención del próximo dígito de hora ASCII
08A4 3C         06031           INR     A                       ;Actualización del mismo
08A5 12         06032           STAX    D                       ;  almacenamiento del mismo
08A6 BE         06033           CMP     M                       ;Comparación con el valor máximo
08A7 C2B008     06034           JNZ     RTC$Clock$Updated       ;No necesita arrastre; por tanto, actual. compl.
08AA 3E30       06035           MVI     A,'0'                   ;Puesta a cero del dígito a ASCII y almacenar
08AC 12         06036           STAX    D                       ;  de nuevo en hora ASCII
08AD C39908     06037           JMP     RTC$Update$Digit        ;Vuelta atrás para el próximo dígito
                06038   ;
                06039   RTC$Clock$Updated:
                06040   RTC$Check$Watchdog:
08B0  2ABF00    06041           LHLD    RTC$Watchdog$Count      ;Obtención de cuenta centinela en curso
08B3 2B         06042           DCX     H                       ;Decremento de la misma
08B4 7C         06043           MOV     A,H                     ;Comprobación de si es 0FFFFH
08B5 B7         06044           ORA     A
08B6 FACB08     06045           JM      RTC$Dog$Not$Set         ;Ha debido ser cero la primera vez
08B9 B5         06046           ORA     L                       ;Comprobación de si ahora es 0
08BA C2C808     06047           JNZ     RTC$Dog$NZ              ;No, no está fuera de tiempo
                06048
```

Figura 8-10. (Continuación.)

```
                06049                                           ;El tiempo de centinela ha transcurrido;
                06050                                           ;  por tanto, "llamar" la rutina adecuada
08BD 21C508     06051           LXI     H,RTC$Watchdog$Return   ;Fijación de la direccion de retorno
08C0 E5         06052           PUSH    H                       ;  preparada para el retorno
08C1 2AC100     06053           LHLD    RTC$Watchdog$Address    ;Transferir control como para CALL
08C4 E9         06054           PCHL
                06055   RTC$Watchdog$Return:                    ;El control volverá aquí desde la rutina
                06056                                           ;  de centinela del usuario
08C5 C3CB08     06057           JMP     RTC$Dog$Not$Set         ;Funciona como cuando el cent. no está activo
                06058
                06059   RTC$Dog$NZ:
08C8 22BF00     06060           SHLD    RTC$Watchdog$Count      ;Almacenamiento de la cuenta decrementada
                06061   RTC$Dog$Not$Set:                        ;  (la cuenta permanece inalterada)
08CB 3E20       06062           MVI     A,IC$EOI                ;Puesta a cero del chip de control de interrup.
08CD D3D8       06063           OUT     IC$OCW2$Port
                06064
08CF D1         06065           POP     D                       ;Realmacenamiento de registro de la pila local
08D0 C1         06066           POP     B
08D1 2A8422     06067           LHLD    PI$User$Stack           ;Conmutar a pila de usuario
08D4 F9         06068           SPHL
08D5 2A8622     06069           LHLD    PI$User$HL              ;Recuperación de registros de usuario
08D8 F1         06070           POP     PSW
08D9 FB         06071           EI                              ;Reactivación de interrupciones
08DA C9         06072           RET
                06073   ;
                06200   ;#
                06201   ;
                06202   ;       Shift HL Right one bit
                06203   ;
                06204   SHLR:
08DB B7         06205           ORA     A               ;Elimina acarreo
08DC 7C         06206           MOV     A,H             ;Obtención de byte MS
08DD 1F         06207           RAR                     ;Bit 7 fijado por el acarreo previo
                06208                                   ;Bit 0 se introduce en el acarreo
08DE 67         06209           MOV     H,A             ;Vuelta a su sitio del byte MS desplazado
08DF 7D         06210           MOV     A,L             ;Obtención del byte LS
08E0 1F         06211           RAR                     ;Bit 7 = bit 0 del byte MS
08E1 6F         06212           MOV     L,A             ;Puesta en resultado
08E2 C9         06213           RET
                06214
                06215   ;
                06300   ;#
                06301   ;       Controladores de alto nivel para disquete
                06302   ;
                06303   ;       Estos controladores realizan las siguientes funciones:
                06304   ;
                06305   ;       SELDSK    Seleccionan un disco específico y devuelven la dirección
                06306   ;                 a la cabecera de parámetros del disco apropiada
                06307   ;       SETTRK    Fijación del número de pista para la próxima lectura o escritura
                06308   ;       SETSEC    Fijación del número de sector para la próxima lectura o escritura
                06309   ;       SETDMA    Fijación de direc. de DMA (lec./esc.) para la próxima lect. o escritura
                06310   ;       SECTRAN   Traduce un número de sector lógico a físico
                06311   ;       HOME      Pone la pista a cero de forma que la siguiente lectura o escritura
                06312   ;                 se encuentre en la pista 0
                06313   ;
                06314   ;       Además, los controladores de alto nivel son responsables de hacer
                06315   ;       que los disquetes flexibles de 5 1/4" que utilizan sectores de 512 bytes
                06316   ;       aparezcan ante CP/M como si utilizaran sectores de 128 bytes. Lo hacen
                06317   ;       utilizando un programa de bloqueo/desbloqueo. Este programa de
                06318   ;       bloqueo/desbloqueo se describe con más detalle más adelante en este
                06319   ;       listado, exactamente antes del propio programa.
                06320   ;
                06321   ;
                06322   ;
                06323   ;       Tablas de parámetros de disco
                06324   ;
                06325   ;       Como se explicó en el capítulo 3, describen las características físicas
                06326   ;       de las unidades de disco. En el BIOS del ejemplo
                06327   ;       existen dos tipos de unidades de disco: estándar 8" una sola cara
                06328   ;       y de una simple densidad y minidisquetes de doble cara, doble
                06329   ;       densidad 5 1/4".
                06330   ;
                06331   ;       Los disquetes estándar 8"  no necesitan utilizar el programa
                06332   ;       de bloqueo/desbloqueo, pero si las unidades de 5 1/4". Por consiguiente,
                06333   ;       se ha prefijado un byte adicional en el bloque de parámetros de disco
                06334   ;       para indicar a los controladores de disco qué disco físico corresponde
```

Figura 8-10. (Continuación.)

```
                        06335   ;       a cada disco lógico y si necesita o no desbloqueo.
                        06336   ;
                        06337   ;
                        06338   ;       Tablas de definición de disco
                        06339   ;
                        06340   ;       Consisten en cabeceras de parámetros, con una entrada por controlador
                        06341   ;       de disco lógico, y bloques de parámetro de disco también
                        06342   ;       con un bloque de parámetros por disco lógico, o el mismo bloque
                        06343   ;       de parámetros para varios discos lógicos.
                        06344   ;
                        06400   ;#
                        06401   ;
                        06402   Disk$Parameter$Headers:                 ;Descrito en el capítulo 3
                        06403   ;
                        06404                           ;Disco lógico A: (disquete de 5 1/4")
08E3 AE09               06405           DW      Floppy$5$Skewtable      ;Tabla de desplazamiento de 5 1/4"
08E5 000000000006406            DW      0,0,0                   ;Reservado para CP/M
08EB B022               06407           DW      Directory$Buffer
08ED 3409               06408           DW      Floppy$5$Parameter$Block
08EF B023               06409           DW      Disk$A$Workarea
08F1 1024               06410           DW      Disk$A$Allocation$Vector
                        06411   ;
                        06412                           ;Disco lógico B: (disquete de 5 1/4")
08F3 AE09               06413           DW      Floppy$5$Skewtable      ;Comparte la misma tabla de desplaz. que A:
08F5 000000000006414            DW      0,0,0                   ;Reservado para CP/M
08FB B022               06415           DW      Directory$Buffer        ;Comparte la misma memoria intermedia que A:
08FD 3409               06416           DW      Floppy$5$Parameter$Block        ;El mismo DPB que A:
08FF D023               06417           DW      Disk$B$Workarea                 ;Area de trabajo particular
0901 2624               06418           DW      Disk$B$Allocation$Vector        ;Vector de situación particular
                        06419   ;
                        06420                           ;Disco lógico C: (flexible de 8")
0903 F609               06421           DW      Floppy$8$Skewtable      ;Tabla de desplazamiento de 8"
0905 000000000006422            DW      0,0,0                   ;Reservado para CP/M
090B B022               06423           DW      Directory$Buffer        ;Comparte la misma memoria intermedia de A:
090D 4409               06424           DW      Floppy$8$Parameter$Block
090F F023               06425           DW      Disk$C$Workarea                 ;Area de trabajo particular
0911 3C24               06426           DW      Disk$C$Allocation$Vector        ;Vector de situación particular
                        06427   ;
                        06428                           ;Disco lógico D: (flexible de 8")
0913 AE09               06429           DW      Floppy$5$Skewtable      ;Comparte la misma tabla de A:
0915 000000000006430            DW      0,0,0                   ;Reservado para CP/M
091B B022               06431           DW      Directory$Buffer        ;Comparte la misma memoria intermedia de A:
091D 4409               06432           DW      Floppy$8$Parameter$Block        ;El mismo DPB de C:
091F 0024               06433           DW      Disk$D$Workarea                 ;Area de trabajo particular
0921 5B24               06434           DW      Disk$D$Allocation$Vector        ;Vector de situación particular
                        06435
                        06436                           ;Disco lógico M: (disco de memoria)
                        06437   M$Disk$DPH:
0923 0000               06438           DW      0                       ;No se necesita desplazamiento
0925 000000000006439            DW      0,0,0                   ;Reservado para CP/M
092B B022               06440           DW      Directory$Buffer
092D 5409               06441           DW      M$Disk$Parameter$Block
092F 0000               06442           DW      0                       ;El disco no puede cambiarse; por tanto,
                        06443                                           ;  no se necesita área de trabajo
0931 7A24               06444           DW      M$Disk$Allocation$Vector
                        06445   ;
                        06446   ;
                        06447   ;       Equivalencias para bloque de parámetros de disco
                        06448   ;
                        06449   ;       Tipos de disco
                        06450   ;
0001 =                  06451   Floppy$5        EQU     1       ;Miniflexible de 5 1/4"
0002 =                  06452   Floppy$8        EQU     2       ;Flexible de 8"  (SC SD)
0003 =                  06453   M$Disk          EQU     3       ;Disco de memoria
                        06454   ;
                        06455   ;       Indicador de bloqueo/desbloqueo
                        06456   ;
0080 =                  06457   Need$Deblocking EQU     1000$0000B      ;Tamaño del sector > 128 bytes
                        06458   ;
                        06600   ;#
                        06601   ;
                        06602   ;       Bloques de parámetros de disco
                        06603   ;
                        06604   ;       Miniflexible de 5 1/4"
                        06605   ;
                        06606                                   ;Prefijado un byte extra para indicar el tipo de disco
```

Figura 8-10. (Continuación.)

```
                        06607                                   ; y el bloqueo requeridos
0933 81                 06608           DB      Floppy$5 + Need$Deblocking
                        06609                                   ;El bloque de parámetros ha sido corregido para
                        06610                                   ; reflejar la nueva distribución de una pista por
                        06611                                   ; lado del disquete, mejor que ver una pista
                        06612                                   ; como ambos lados en una posición determinada
                        06613                                   ; de la cabeza.
                        06614                                   ;También se ha ajustado para reflejar
                        06615                                   ; una "nueva" pista más que se utiliza
                        06616                                   ; para la imagen del CP/M,
                        06617                                   ; con el cambio resultante en el número
                        06618                                   ; de bloques y el número de pistas
                        06619                                   ; reservadas.
                        06620   Floppy$5$Parameter$Block:
0934 2400               06621           DW      36              ;Sectores de 128 bytes por pista
0936 04                 06622           DB      4               ;Desplazamiento de bloque
0937 0F                 06623           DB      15              ;Máscara de bloque
0938 01                 06624           DB      1               ;Máscara de extensión
0939 AB00               06625           DW      171             ;Número máximo de bloques
093B 7F00               06626           DW      127             ;Número de entradas del directorio - 1
093D C0                 06627           DB      1100$0000B      ;Mapa de bits para reservar un bloque para
093E 00                 06628           DB      0000$0000B      ; el directorio de ficheros
093F 2000               06629           DW      32              ;Tamaño del área de trabajo de disco cambiado
0941 0300               06630           DW      3               ;Número de pistas antes del directorio
                        06631   ;
                        06632   ;
                        06633   ;       Estándar flexible de 8"
                        06634                                   ;Prefijado byte extra para DPB para esta
                        06635                                   ; versión del BIOS
0943 02                 06636           DB      Floppy$8        ;Indica el tipo de disco y el hecho de que no
                        06637                                   ; se requiere desbloqueo
                        06638   Floppy$8$Parameter$Block:
0944 1A00               06639           DW      26              ;Sectores por pista
0946 03                 06640           DB      3               ;Desplazamiento de bloque
0947 07                 06641           DB      7               ;Máscara de bloque
0948 00                 06642           DB      0               ;Máscara de extensión
0949 F200               06643           DW      242             ;Número máximo de bloques
094B 3F00               06644           DW      63              ;Número de entradas del directorio - 1
094D C0                 06645           DB      1100$0000B      ;Mapa de bits para reservar dos bloques para
094E 00                 06646           DB      0000$0000B      ; el directorio de fichero
094F 1000               06647           DW      16              ;Tamaño del área de trabajo de disco cambiado
0951 0200               06648           DW      2               ;Número de pistas antes del directorio
                        06649   ;
                        06650   ;       M$Disk
                        06651   ;
                        06652                                   ;El M$Disk supone que hay disponibles 4 x 48K bancos
                        06653                                   ; de memoria. La tabla siguiente describe
                        06654                                   ; el disco como si tuviera 8 pistas: dos
                        06655                                   ; de ellas para el banco de memoria
                        06656                                   ; con cada pista de 192 sectores
                        06657                                   ; de 128 bytes.
                        06658                                   ; El número de pista dividido por dos se utilizará
                        06659                                   ; para seleccionar el banco.
0953 03                 06660           DB      M$Disk          ;El tipo es M$Disk, no desbloqueo
                        06661   M$Disk$Parameter$Block:
0954 C000               06662           DW      192             ;Sectores por "pista". Cada pista es
                        06663                                   ; 24K de memoria
0956 03                 06664           DB      3               ;Desplazamiento de bloque (situación del byte 1024)
0957 07                 06665           DB      7               ;Máscara de bloque
0958 00                 06666           DB      0               ;Máscara de extensión
0959 C000               06667           DW      192             ;Número máximo de bloque
095B 3F00               06668           DW      63              ;Número de entradas del directorio - 1
095D C0                 06669           DB      1100$0000B      ;Mapa de bits para reservar dos bloques para
095E 00                 06670           DB      0000$0000B      ; el directorio de ficheros
095F 0000               06671           DW      0               ;El disco no puede cambiarse; por tanto,
                        06672                                   ; no existe área de trabajo
0961 0000               06673           DW      0               ;Ninguna pista reservada
                        06674   ;
0004 =                  06675   Number$of$Logical$Disks         EQU     4
                        06676   ;
                        06800   ;*
                        06801   ;
                        06802   SELDSK:         ;Disco seleccionado en registro C
                        06803                           ;C = 0 para unidad A, 1 para B, etc.
                        06804                           ;Retorno de la dirección de la cabecera de
                        06805                           ; parámetros de disco adecuada en HL, o 0000H
```

Figura 8-10. (Continuación.)

```
                06806                           ;  si el disco seleccionado no existe
                06807                           ;
0963 210000     06808           LXI     H,0             ;Supone un error
0966 79         06809           MOV     A,C             ;Comprobación de si es disco solicitado válido
                06810
0967 FE0C       06811           CPI     'M' - 'A'       ;Comprobación de si es disco de memoria
0969 CA9509     06812           JZ      SELDSK$M$Disk   ;Sí
                06813
096C FE04       06814           CPI     Number$of$Logical$Disks
096E D0         06815           RNC                     ;Retorno si > número máximo de discos
                06816   ;
096F 322D0A     06817           STA     Selected$Disk   ;Almacenamiento número de disco seleccionado
                06818                                   ;Fijación para volver a la dirección DPH
0972 6F         06819           MOV     L,A             ;Componer disco en valor palabra
0973 2600       06820           MVI     H,0
                06821                                   ;Cálculo del desplazamiento de la tabla cabecera
                06822                                   ;  de parámetros del disco multiplicando por la
                06823                                   ;  longitud de cabecera de parámetros (16 bytes)
0975 29         06824           DAD     H               ;*2
0976 29         06825           DAD     H               ;*4
0977 29         06826           DAD     H               ;*8
0978 29         06827           DAD     H               ;*16
0979 11E308     06828           LXI     D,Disk$Parameter$Headers   ;Obtención de dirección base
097C 19         06829           DAD     D               ;DE -> DPH adecuado
097D E5         06830           PUSH    H               ;Almacenamiento de la dirección DPH
                06831   ;
                06832                                   ;Acceso al bloque de parámetro de disco para extraer
                06833                                   ;  el byte especial de prefijo
                06834                                   ;  que identifica el tipo de disco y si se
                06835                                   ;  requiere el desbloqueo
                06836                                   ;
097E 110A00     06837           LXI     D,10            ;Obtención del desplazamiento del apuntador DPB en DPH
0981 19         06838           DAD     D               ;DE -> dirección DPB en DPH
0982 5E         06839           MOV     E,M             ;Obtención dirección DPB en DE
0983 23         06840           INX     H
0984 56         06841           MOV     D,M
0985 EB         06842           XCHG                    ;DE -> DPB
                06843
                06844   SELDSK$Set$Disk$Type:
0986 2B         06845           DCX     H               ;DE -> byte de prefijo
0987 7E         06846           MOV     A,M             ;Obtención del byte de prefijo
0988 E60F       06847           ANI     0FH             ;Aislamiento del tipo de disco
098A 32360A     06848           STA     Selected$Disk$Type      ;Almac. para uso en control. de bajo nivel
098D 7E         06849           MOV     A,M             ;Obtención de otra copia del byte de prefijo
098E E680       06850           ANI     Need$Deblocking         ;Aislamiento indicador de desbloqueo
0990 32350A     06851           STA     Selected$Disk$Deblock   ;Almac. para uso en controlador de bajo nivel
0993 E1         06852           POP     H               ;Recuperación de apuntador DPH
0994 C9         06853           RET
                06854   ;
                06855   SELDSK$M$Disk:                          ;M$Disk seleccionado
0995 212309     06856           LXI     H,M$Disk$DPH            ;Retorno de la cabecera de parám. correcta
0998 C38609     06857           JMP     SELDSK$Set$Disk$Type    ;Reanudar proceso normal
                06858   ;
                07000   ;#
                07001   ;
                07002   ;       Fija la pista lógica para la próxima lectura o escritura
                07003   ;
                07004   SETTRK:
099B 60         07005           MOV     H,B             ;En entrada pista seleccionada en BC
099C 69         07006           MOV     L,C
099D 222E0A     07007           SHLD    Selected$Track  ;Almacenamiento para controlador de bajo nivel
09A0 C9         07008           RET
                07009   ;
                07100   ;#
                07101   ;
                07102   ;       Fijación del sector lógico para la próxima lectura o escritura
                07103   ;
                07104   ;
                07105   SETSEC:                         ;En entrada sector lógico en C
09A1 79         07106           MOV     A,C
09A2 32300A     07107           STA     Selected$Sector ;Almacenamiento para el controlador de bajo nivel
09A5 C9         07108           RET
                07109   ;
                07200   ;#
                07201   ;
                07202   ;       Fijación dirección DMA del disco (E/S) para la próxima lectura o escritura
```

Figura 8-10. (Continuación.)

```
                    07203   ;
09A6 0000           07204   DMA$Address:    DW      0               ;Dirección DMA
                    07205   ;
                    07206   SETDMA:                                 ;En entrada, dirección en BC
09A8 69             07207           MOV     L,C                     ;Desplazar a HL para almacenamiento
09A9 60             07208           MOV     H,B
09AA 22A609         07209           SHLD    DMA$Address             ;Almacenamiento para controlador de bajo nivel
09AD C9             07210           RET
                    07211   ;
                    07300   ;#
                    07301   ;
                    07302   ;       Traducción número de sector lógico a físico
                    07303   ;
                    07304   ;       Tablas de traducción de sector
                    07305   ;       Estas tablas están indexadas utilizando el número de sector lógico,
                    07306   ;       y contiene el número de sector físico correspondiente.
                    07307   ;
                    07308   Floppy$5$Skewtable:     ;Cada sector físico contiene cuatro
                    07309                           ;sectores de 128 bytes
                    07310           ;       Físico 128b     Lógico 128b     Físico 512 byte
09AE 00010203       07311           DB      00,01,02,03     ;00,01,02,03        0   )
09B2 10111213       07312           DB      16,17,18,19     ;04,05,06,07        4   )
09B6 20212223       07313           DB      32,33,34,35     ;08,09,10,11        8   )
09BA 0C0D0E0F       07314           DB      12,13,14,15     ;12,13,14,15        3   ) Cabeza
09BE 1C1D1E1F       07315           DB      28,29,30,31     ;16,17,18,19        7   )   0
09C2 08090A0B       07316           DB      08,09,10,11     ;20,21,22,23        2   )
09C6 18191A1B       07317           DB      24,25,26,27     ;24,25,26,27        6   )
09CA 04050607       07318           DB      04,05,06,07     ;28,29,30,31        1   )
09CE 14151617       07319           DB      20,21,22,23     ;32,33,34,35        5   )
                    07320   ;
09D2 24252627       07321           DB      36,37,38,39     ;36,37,38,39        0   ]
09D6 34353637       07322           DB      52,53,54,55     ;40,41,42,43        4   ]
09DA 44454647       07323           DB      68,69,70,71     ;44,45,46,47        8   ]
09DE 30313233       07324           DB      48,49,50,51     ;48,49,50,51        3   ] cabeza
09E2 40414243       07325           DB      64,65,66,67     ;52,53,54,55        7   ]   1
09E6 2C2D2E2F       07326           DB      44,45,46,47     ;56,57,58,59        2   ]
09EA 3C3D3E3F       07327           DB      60,61,62,63     ;60,61,62,63        6   ]
09EE 28292A2B       07328           DB      40,41,42,43     ;64,65,66,67        1   ]
09F2 38393A3B       07329           DB      56,57,58,59     ;68,69,70,71        5   ]
                    07330   ;
                    07331   ;
                    07332   Floppy$8$Skewtable:     ;Controlador estándar de 8"
                    07333           ;       01,02,03,04,05,06,07,08,09,10    Sectores lógicos
09F6 01070D131907334            DB      01,07,13,19,25,05,11,17,23,03   ;Sectores físicos
                    07335           ;
                    07336           ;       11,12,13,14,15,16,17,18,19,20    Sectores lógicos
0A00 090F15020807337            DB      09,15,21,02,08,14,20,26,06,12   ;Sectores físicos
                    07338           ;
                    07339           ;       21,22,23,24,25,26        Sectores lógicos
0A0A 1218040A1007340            DB      18,24,04,10,16,22       ;Sectores físicos
                    07341
                    07400   ;#
                    07401   ;
                    07402   SECTRAN:                        ;Traducir sector lógico a físico
                    07403                                   ;En entrada, BC = número de sector lógico
                    07404                                   ;               DE -> tabla de desplazamiento adecuada
                    07405                                   ;
                    07406                                   ;En salida, HL = número de sector físico
0A10 EB             07407           XCHG                            ;HL -> base de tabla de desplazamiento
0A11 09             07408           DAD     B                       ;Añadir al número de sector lógico
0A12 6E             07409           MOV     L,M                     ;Obtención número de sector físico
0A13 2600           07410           MVI     H,0                     ;Poner en un valor de 16 bits
0A15 C9             07411           RET
                    07412   ;
                    07500   ;#
                    07501   ;
                    07502   ;
                    07503   HOME:                           ;Situar el disco lógico seleccionado en la pista 0
                    07504                                   ;Antes de hacerlo, deberá comprobarse si la memoria
                    07505                                   ;  intermedia del disco físico tiene información en ella que
                    07506                                   ;  haya de ser escrita. Esto se indica por medio de un
                    07507                                   ;  indicador, Must$Write$Buffer, que está situado
                    07508                                   ;  en el programa de desbloqueo.
                    07509                                   ;
0A16 3A2C0A         07510           LDA     Must$Write$Buffer       ;Comprobación de si la memoria intermedia
0A19 B7             07511           ORA     A                       ;  física ha de ser escrita en un disco
```

Figura 8-10. (Continuación.)

```
0A1A C2200A     07512            JNZ     HOME$No$Write
0A1D 322B0A     07513            STA     Data$In$Disk$Buffer    ;No, quiere decir que la memoria intermedia
                07514                                           ;  está ahora desocupada
                07515    HOME$No$Write:
0A20 0E00       07516            MVI     C,0                    ;Fijar en la pista 0 (lógicamente, no sucede
0A22 CD9B09     07517            CALL    SETTRK                 ;  ninguna operación práctica del disco)
0A25 C9         07518            RET
                07519
                07520    ;
                07600    ;*
                07601    ;       Los datos escritos o leídos por la unidad de disquete se transfieren
                07602    ;       por medio de una memoria intermedia física que es una pista completa
                07603    ;       en longitud, 9 * 512 bytes. Se declara al final del BIOS,
                07604    ;       y tiene un pequeño programa de inicialización "escondida" en él.
                07605    ;
                07606    ;       El programa de bloqueo/desbloqueo intenta minimizar la cantidad
                07607    ;       de entradas/salidas del disco real almacenando el disco y la pista
                07608    ;       en curso que residen en la memoria intermedia física.
                07609    ;       Si sucede una petición de lectura en un "sector" CP/M de 128 bytes
                07610    ;       y si está en la memoria intermedia física, no se realizará
                07611    ;       ningún acceso al disco a menos que el BDOS indique
                07612    ;       que está escribiendo en el directorio. La escritura
                07613    ;       en el directorio provoca una escritura inmediata
                07614    ;       en el disco de la pista entera en la memoria
                07615    ;       intermedia física.
                07616    ;
                07617    ;
0800 =          07618    Allocation$Block$Size   EQU     2048
0009 =          07619    Physical$Sec$Per$Track  EQU     9       ;Ajustada para reflejar que una
                07620                                           ;  "nueva" pista es sólo una cara
                07621                                           ;  del disco
0200 =          07622    Physical$Sector$Size    EQU     512     ;Este es el tamaño del sector real para los
                07623                                           ;  disquetes 5 1/4"
                07624                                           ;Los disquetes de 8"  y los discos de memoria
                07625                                           ;  utilizan sectores de 128 bytes
                07626                                           ;Declaración de la memoria intermedia de disco
                07627                                           ;  físico para los disquetes de 5 1/4"
0004 =          07628    CPM$Sec$Per$Physical    EQU     Physical$Sector$Size/128
0024 =          07629    CPM$Sec$Per$Track       EQU     CPM$Sec$Per$Physical*Physical$Sec$Per$Track
1200 =          07630    Bytes$Per$Track         EQU     Physical$Sec$Per$Track*Physical$Sector$Size
0003 =          07631    Sector$Mask             EQU     CPM$Sec$Per$Physical-1
0002 =          07632    Sector$Bit$Shift        EQU     2               ;LOG2(CPM$Sec$Per$Physical)
                07633    ;
                07634                            ;Estos son los valores transmitidos por el BDOS cuando llama
                07635                            ;  a la operación de escritura.
                07636                            ;El asignado/no asignado indica si el BDOS desea
                07637                            ;  escribir en un bloque no asignado (sólo indica esto
                07638                            ;  para la primera escritura de sector de 128 bytes)
                07639                            ;  o en un bloque que ha sido ya asignado
                07640                            ;  a un fichero.
                07641                            ;El BDOS indica también si desea escribir en el directorio
                07642                            ;  del fichero.
                07643                            ;
0000 =          07644    Write$Allocated         EQU     0
0001 =          07645    Write$Directory         EQU     1
0002 =          07646    Write$Unallocated       EQU     2       ;<== Ignorado para la mem. int. de pista
                07647    ;
0A26 00         07648    Write$Type:             DB      0       ;Contiene el tipo de escritura indicado
                07649                                           ;  por el BDOS
                07650    ;
                07651    ;
                07652    In$Buffer$Dk$Trk:                       ;Variables para sector físico en memoria
                07653                                            ;  Disk$Buffer
0A27 00         07654    In$Buffer$Disk:         DB      0       ;) Son movidos y comparados como grupo,
0A28 0000       07655    In$Buffer$Track:        DW      0       ;) de forma que no se alteren
                07656                                           ;  estas líneas
0A2A 00         07657    In$Buffer$Disk$Type:    DB      0       ;Tipo de disco para sector en la mem. int.
                07658    ;
0A2B 00         07659    Data$In$Disk$Buffer:    DB      0       ;Si es distinto de cero, la memoria intermedia
                07660                                           ;  del disco tiene datos del disco en ella
0A2C 00         07661    Must$Write$Buffer:      DB      0       ;No cero cuando los datos se han escrito
                07662                                           ;  en el Disk$Buffer pero aún no se han
                07663                                           ;  escrito en el disco
                07664    ;
                07665    Selected$Dk$Trk:                        ;Variables para el disco, la pista y el sector
                07666                                            ;seleccionados (selec. por SELDSK, SETTRK y SETSEC)
```

Figura 8-10. (Continuación.)

```
0A2D 00          07667   Selected$Disk:          DB      0       ;) Se mueven y se comparan como un grupo de
0A2E 0000        07668   Selected$Track:         DW      0       ;) forma que no se altere el orden
                 07669
0A30 00          07670   Selected$Sector:        DB      0       ;No forma parte del grupo, pero se
                 07671                                           ;  necesita aquí
0A31 00          07672   Selected$Physical$Sector: DB    0       ;Sector físico seleccionado derivado del
                 07673                                           ;  sector (CP/M) seleccionado mediante
                 07674                                           ;  desplazamiento a la derecha del número de
                 07675                                           ;  bits especificado por Sector$Bit$Shift
                 07676
                 07677   ;
                 07678
0A32 00          07679   Disk$Error$Flag:        DB      0       ;No cero para indicar un error que podrá
                 07680                                           ;  no ser recuperado por los
                 07681                                           ;  controladores del disco. El BDOS llevará a
                 07682                                           ;  la salida un mensaje de "Sector defectuoso"
0A33 00          07683   Disk$Hung$Flag:         DB      0       ;No cero si ocurre un tiempo excedido
                 07684                                           ;  del centinela
0258 =           07685   Disk$Timer              EQU     600     ;Número de 16,66 como pulsos de reloj para un
                 07686                                           ;  exceso de tiempo de 10 segundos
                 07687   ;
                 07688                                   ;Indicadores utilizados dentro del prog. de desbloqueo
                 07689
0A34 00          07690   Read$Operation:         DB      0       ;No cero cuando hay que leer un sector
                 07691                                           ;  de 128 bytes del CP/M
0A35 00          07692   Selected$Disk$Deblock:  DB      0       ;No cero cuando el disco seleccionado necesita
                 07693                                           ;  desbloqueo (fijado en SELDSK)
0A36 00          07694   Selected$Disk$Type:     DB      0       ;Indica el disco flexible de 8" o de 5 1/4" o
                 07695                                           ;  M$Disk de seleccionado (fijado en SELDSK)
                 07696   ;
                 07800   ;#
                 07801   ;
                 07802   ;       Lee en el sector de 128 bytes del CP/M especificado por llamadas previas
                 07803   ;       de selección de disco. Fija la pista y el sector. El sector se leerá
                 07804   ;       en la dirección especificada en la llamada previa de Set DMA.
                 07805   ;
                 07806   ;       Si se lee de una unidad de disco utilizando sectores mayores
                 07807   ;       de 128 bytes, el programa de desbloqueo se utilizará para "desempaquetar"
                 07808   ;       un sector de 128 bytes del sector físico.
                 07809   READ:
0A37 3A350A      07810           LDA     Selected$Disk$Deblock   ;Comprobación de si es necesario desbloqueo
0A3A B7          07811           ORA     A                       ;  (indicador fijado en la llamada a SELDSK)
0A3B CA2F0B      07812           JZ      Read$No$Deblock         ;No, utilizar no desbloqueado normales
                 07813
                 07814                                   ;El algoritmo de desbloqueo utilizado es tal
                 07815                                   ;  que una operación de lectura puede ser vista
                 07816                                   ;  hasta que los datos en curso se transfieren
                 07817                                   ;  como si fuese la primera lectura
                 07818                                   ;  a un bloque no asignado
0A3E 3E01        07819           MVI     A,1                     ;Indica que debe realizarse prácticamente
0A40 32340A      07820           STA     Read$Operation          ;  una lectura
                 07821
0A43 3E00        07822           MVI     A,Write$Allocated       ;Simula programa de desbloqueo donde se desee
0A45 32260A      07823           STA     Write$Type              ;  que éste sea una escritura
                 07824                                           ;  en un bloque asignado
0A48 C35C0A      07825           JMP     Perform$Read$Write      ;Utiliza programa común para ejecutar la lec.
                 07826   ;
                 07900   ;#
                 07901   ;       Escritura de sector de 128 bytes de la dirección DMA en curso,
                 07902   ;       al disco, pista y sector seleccionados previamente.
                 07903   ;
                 07904   ;       Al llegar aquí, el BDOS tendrá fijado el registro C para indicar
                 07905   ;       si esta operación de escritura es para un bloque ya asignado
                 07906   ;       (lo que significa que puede ser necesaria una lectura previa del sector)
                 07907   ;       o para el directorio (caso en el cual los datos serán escritos
                 07908   ;       inmediatamente en el disco).
                 07909   ;
                 07910   ;       Sólo la escritura en el directorio tiene lugar inmediatamente. En todos
                 07911   ;       los demás casos, los datos se trasladarán de la dirección DMA a la memoria
                 07912   ;       intermedia del disco, y serán escritos sólo cuando las circunstancias
                 07913   ;       obliguen a la transferencia. El número de operaciones físicas
                 07914   ;       de disco, por tanto, puede reducirse considerablemente.
                 07915   ;
                 07916   WRITE:
0A4B 3A350A      07917           LDA     Selected$Disk$Deblock   ;Comprobación de si se requiere desbloqueo
0A4E B7          07918           ORA     A                       ;  (indicador fijado en la llamada SELDSK)
```

Figura 8-10. (Continuación.)

```
0A4F CA2A0B   07919            JZ      Write$No$Deblock
              07920
0A52 AF       07921            XRA     A                    ;Indica que se requiere una operación de
0A53 32340A   07922            STA     Read$Operation       ;  escritura (es decir, NO una lectura)
0A56 79       07923            MOV     A,C                  ;Salvar tipo de escritura del BDOS, pero
0A57 E601     07924            ANI     1                    ;  distinguiendo únicamente entre
              07925                                         ;  escritura en el bloque o escritura
0A59 32260A   07926            STA     Write$Type           ;  en directorio
              07927    ;
              07928    ;
              08000    ;#
              08001    ;
              08002    Perform$Read$Write:     ;Programa común para ejecutar tanto lecturas
              08003                           ;  como escrituras de sectores de 128 bytes
0A5C AF       08004            XRA     A              ;Suponiendo que no ocurran errores de
0A5D 32320A   08005            STA     Disk$Error$Flag ;  disco
              08006
0A60 3A300A   08007            LDA     Selected$Sector ;Conversión del sector seleccionado de 128 bytes
0A63 1F       08008            RAR                     ;  en sector físico dividiendo por 4
0A64 1F       08009            RAR
0A65 E63F     08010            ANI     3FH             ;Elimina cualquier bit no deseado
0A67 32310A   08011            STA     Selected$Physical$Sector
              08012                                    ;
0A6A 212B0A   08013            LXI     H,Data$In$Disk$Buffer   ;Comprobación de si la memoria intermedia del
0A6D 7E       08014            MOV     A,M                     ;  disco ya tiene datos en ella
0A6E 3601     08015            MVI     M,1                     ;(Indica incondicionalmente que la memoria
              08016                                            ;  intermedia tiene ahora datos en ella)
0A70 B7       08017            ORA     A                       ;¿Tiene datos efectivamente en ella?
0A71 CA870A   08018            JZ      Read$Track$into$Buffer  ;No, procede leer una pista física en la
              08019                                            ;  memoria intermedia
              08020                                    ;
              08021                                    ;La memoria intermedia tiene en su interior una pista
              08022                                    ;  física. Comprobación de si es la correcta
              08023                                    ;
0A74 11270A   08024            LXI     D,In$Buffer$Dk$Trk      ;Comprobación de si la pista de la mem. int.
0A77 212D0A   08025            LXI     H,Selected$Dk$Trk       ;  es la misma que la selec. previamente
0A7A CDE10A   08026            CALL    Compare$Dk$Trk          ;Compara SOLO disco y pista
0A7D CA910A   08027            JZ      Track$In$Buffer         ;Si, está ya en la memoria intermedia
              08028
              08029                                          ;No, habrá que leerla en los contenidos
              08030                                          ;  en curso de la memoria intermedia
0A80 3A2C0A   08031            LDA     Must$Write$Buffer       ;Comprobación de si la memoria intermedia
0A83 B7       08032            ORA     A                       ;  tiene datos que deban escribirse antes
0A84 C4E50B   08033            CNZ     Write$Physical          ;Si, escribirse
              08034    ;
              08035    Read$Track$into$Buffer:
0A87 CDCE0A   08036            CALL    Set$In$Buffer$Dk$Trk    ;Fija en la memoria intermedia variables
              08037                                          ;  del disco seleccionado y pista
              08038                                          ;  para reflejar cuál de éstas está ahora
              08039                                          ;  en la memoria intermedia
0A8A CDEA0B   08040            CALL    Read$Physical           ;Lee la pista en la memoria intermedia
0A8D AF       08041            XRA     A                       ;Pone a cero el indicador para reflejar los
0A8E 322C0A   08042            STA     Must$Write$Buffer       ;  contenidos de la memoria intermedia
              08043    ;
              08044    Track$In$Buffer:                ;La pista seleccionada y disco están
              08045                                    ;  ya en la memoria intermedia
              08046                                    ;Conversión del sector seleccionado
              08047                                    ;  de 128 bytes CP/M en una dirección relativa
              08048                                    ;  bajo la memoria intermedia
0A91 3A300A   08049            LDA     Selected$Sector ;Obtención del número de sector seleccionado
0A94 6F       08050            MOV     L,A             ;Multiplica por 128 desplazando valor de 16 bits
0A95 2600     08051            MVI     H,0             ;7 bits a la izquierda
0A97 29       08052            DAD     H               ;* 2
0A98 29       08053            DAD     H               ;* 4
0A99 29       08054            DAD     H               ;* 8
0A9A 29       08055            DAD     H               ;* 16
0A9B 29       08056            DAD     H               ;* 32
0A9C 29       08057            DAD     H               ;* 64
0A9D 29       08058            DAD     H               ;* 128
              08059    ;
0A9E 11A40F   08060            LXI     D,Disk$Buffer   ;Obtención de dirección de base de mem. int. de disco
0AA1 19       08061            DAD     D               ;Suma al número de sector * 128
              08062                                    ;HL -> dirección de comienzo del número de sector de
              08063                                    ;  128 bytes en la memoria intermedia de disco
0AA2 EB       08064            XCHG                    ;DE -> sector en la memoria intermedia de disco
0AA3 2AA609   08065            LHLD    DMA$Address     ;Obtención de direc. DMA fijada en la llamada SETDMA
```

Figura 8-10. (Continuación.)

```
0AA6 EB          08066            XCHG                        ;Supone una operación de lectura, de forma que
                 08067                                        ;  DE -> dirección de DMA
                 08068                                        ;  HL -> sector en la memoria intermedia de disco
0AA7 0E10        08069            MVI     C,128/8             ;A causa del método rápido utilizado para introducir
                 08070                                        ;  y sacar datos de la memoria intermedia
                 08071                                        ;  del disco (ocho bytes movidos por bucle
                 08072                                        ;  de iteración), el contador necesita
                 08073                                        ;  sólo ser 1/8 de lo normal
                 08074                                        ;En este punto,
                 08075                                        ;  C = contador de bucle
                 08076                                        ;  DE -> dirección de DMA
                 08077                                        ;  HL -> sector en la memoria intermedia del disco
0AA9 3A340A      08078            LDA     Read$Operation      ;Determina si los datos deben ser sacados de la
0AAC B7          08079            ORA     A                   ;  memoria intermedia (lectura) o introducidos
0AAD C2B50A      08080            JNZ     Buffer$Move         ;  en la memoria intermedia (escritura)
                 08081                                        ;Escribe en la memoria intermedia
                 08082                                               ;(A debe ser 0 obtenido aquí)
0AB0 3C          08083            INR     A                          ;Fija indicador para forzar una escritura de
0AB1 322C0A      08084            STA     Must$Write$Buffer          ;  la mem. int. del disco más adelante
0AB4 EB          08085            XCHG                               ;Hace DE -> sector en la mem. int. del disco
                 08086                                               ;     HL -> dirección de DMA
                 08087    ;
                 08088    ;
                 08089    Buffer$Move:
0AB5 CDF80A      08090            CALL    Move$8              ;Transferencia de 8 bytes * C veces
                 08091                                        ;  de (HL) a (DE)
                 08092                                        ;
0AB8 3A260A      08094            LDA     Write$Type          ;Si escribe en el directorio, escribir
0ABB FE01        08095            CPI     Write$Directory     ;  inmediatamente la memoria intermedia
0ABD 3A320A      08096            LDA     Disk$Error$Flag     ;Obt. del indicador de error en caso de esc./lec. retar.
0AC0 C0          08097            RNZ                         ;Retorno si hay algún retraso en lectura o escritura
                 08098                                        ;
0AC1 B7          08099            ORA     A                   ;Comprobación de si han ocurrido errores de disco
0AC2 C0          08100            RNZ                         ;Sí, abandona el intento de escribir en el directorio
                 08101                                        ;
0AC3 AF          08102            XRA     A                   ;Borrado de indicador de que la memoria intermedia
0AC4 322C0A      08103            STA     Must$Write$Buffer       ;  indica debe ser escrita
0AC7 CDE50B      08104            CALL    Write$Physical      ;Escritura de la memoria intermedia en la pista física
0ACA 3A320A      08105            LDA     Disk$Error$Flag     ;Retorno de indicador de error al llamador
0ACD C9          08106            RET
                 08107    ;
                 08108    ;
                 08109    ;
                 08110    Set$In$Buffer$Dk$Trk:                       ;Indica el disco y la pista seleccionados que
                 08111                                                ;  residen ahora en la memoria intermedia
0ACE 3A2D0A      08112            LDA     Selected$Disk
0AD1 32270A      08113            STA     In$Buffer$Disk
                 08114
0AD4 2A2E0A      08115            LHLD    Selected$Track
0AD7 22280A      08116            SHLD    In$Buffer$Track
                 08117
0ADA 3A360A      08118            LDA     Selected$Disk$Type      ;Refleja también el tipo de disco
0ADD 322A0A      08119            STA     In$Buffer$Disk$Type
                 08120
0AE0 C9          08121            RET
                 08122    ;
                 08123    ;
                 08124    Compare$Dk$Trk:                     ;Compara precisamente disco y pista
                 08125                                        ;  apuntados por DE y HL
0AE1 0E03        08126            MVI     C,3                 ;Disco (1), pista (2)
                 08127    Compare$Dk$Trk$Loop:
0AE3 1A          08128            LDAX    D                   ;Obtención comparado
0AE4 BE          08129            CMP     M                   ;Comparación con comparando
0AE5 C0          08130            RNZ                         ;Abandono de comparación si se encuentra una desig.
0AE6 13          08131            INX     D                   ;Actualización del indicador del comparado
0AE7 23          08132            INX     H                   ;Actualización del indicador del comparando
0AE8 0D          08133            DCR     C                   ;Decrementar en la cuenta de bucle
0AE9 C8          08134            RZ                          ;Retorno (con indicador puesto a cero)
0AEA C3E30A      08135            JMP     Compare$Dk$Trk$Loop
                 08136    ;
                 08137    ;
                 08138    Move$Dk$Trk:                        ;Transferencia de las variables disco y pista
                 08139                                        ;  apuntadas por HL a las apuntadas
                 08140                                        ;  por DE
```

Figura 8-10. (Continuación.)

```
0AED 0E03       08141           MVI     C,3             ;Disco (1), pista (2)
                08142   Move$Dk$Trk$Loop:
0AEF 7E         08143           MOV     A,M             ;Obtención del byte fuente
0AF0 12         08144           STAX    D               ;Almacenamiento en destino
0AF1 13         08145           INX     D               ;Actualización de indicadores
0AF2 23         08146           INX     H
0AF3 0D         08147           DCR     C               ;Decremento de la cuenta de bytes
0AF4 C8         08148           RZ                      ;Retorno si todos los bytes han sido transferidos
0AF5 C3EF0A     08149           JMP     Move$Dk$Trk$Loop
                08150   ;
                08300   ;*
                08301   ;
                08302   ;       Transferencia de ocho bytes
                08303   ;
                08304   ;       Esta rutina transfiere ocho bytes en un bloque, C veces, de (HL)
                08305   ;       a (DE). Utiliza un programa "de salto-a-través" para acelerar
                08306   ;       la ejecución.
                08307   ;
                08308   ;       Parámetros de entrada
                08309   ;
                08310   ;               C = número de bloques de 8 bytes a transferir
                08311   ;               DE -> dirección de destino
                08312   ;               HL -> dirección de fuente
                08313   ;
                08314   Move$8:
0AF8 7E         08315           MOV     A,M             ;Obtención de byte de la fuente
0AF9 12         08316           STAX    D               ;Introducción en destino
0AFA 13         08317           INX     D               ;Actualización indicadores
0AFB 23         08318           INX     H
0AFC 7E         08319           MOV     A,M             ;Obtención de byte de la fuente
0AFD 12         08320           STAX    D               ;Introducción en destino
0AFE 13         08321           INX     D               ;Actualización indicadores
0AFF 23         08322           INX     H
0B00 7E         08323           MOV     A,M             ;Obtención de byte de la fuente
0B01 12         08324           STAX    D               ;Introducción en destino
0B02 13         08325           INX     D               ;Actualización indicadores
0B03 23         08326           INX     H
0B04 7E         08327           MOV     A,M             ;Obtención de byte de la fuente
0B05 12         08328           STAX    D               ;Introducción en destino
0B06 13         08329           INX     D               ;Actualización indicadores
0B07 23         08330           INX     H
0B08 7E         08331           MOV     A,M             ;Obtención de byte de la fuente
0B09 12         08332           STAX    D               ;Introducción en destino
0B0A 13         08333           INX     D               ;Actualización indicadores
0B0B 23         08334           INX     H
0B0C 7E         08335           MOV     A,M             ;Obtención de byte de la fuente
0B0D 12         08336           STAX    D               ;Introducción en destino
0B0E 13         08337           INX     D               ;Actualización indicadores
0B0F 23         08338           INX     H
0B10 7E         08339           MOV     A,M             ;Obtención de byte de la fuente
0B11 12         08340           STAX    D               ;Introducción en destino
0B12 13         08341           INX     D               ;Actualización indicadores
0B13 23         08342           INX     H
0B14 7E         08343           MOV     A,M             ;Obtención de byte de la fuente
0B15 12         08344           STAX    D               ;Introducción en destino
0B16 13         08345           INX     D               ;Actualización indicadores
0B17 23         08346           INX     H
                08347
0B18 0D         08348           DCR     C               ;Decrementa en el contador de bucle
0B19 C2F80A     08349           JNZ     Move$8          ;Repetición hasta completar operación
0B1C C9         08350           RET
                08351
                08352   ;
                08500   ;*
                08501   ;
                08502   ;       Introducción a los controladores físicos de disco en este sistema de cálculo
                08503   ;
                08504   ;       Hay dos controladores de disco "inteligentes" en este sistema,
                08505   ;       uno para las unidades de disquete de 8" y otro para las unidades
                08506   ;       de 5 1/4".
                08507   ;
                08508   ;       Los controladores están "conectados físicamente" al monitor en ciertos
                08509   ;       lugares en la memoria para detectar cuándo debe realizarse alguna
                08510   ;       operación de disco. Monitor-controlador de 8"  posición 0040H.
                08511   ;       Monitor-controlador de 5 1/4"  posición 0045H. Son los llamados bytes de control
                08512   ;       de disco. Si el bit más significativo del byte de control de disco está a 1,
                08513   ;       el controlador examinará la palabra que sigue a los bytes de
```

Figura 8-10. (Continuación.)

```
                        08514   ;       control respectivos. Esta palabra debe contener la dirección
                        08515   ;       de una tabla de control de disco válida que especifique
                        08516   ;       la operación exacta a realizar.
                        08517   ;
                        08518   ;       Una vez que la operación ha terminado, el controlador pone
                        08519   ;       su byte de control a 00H, lo que indica al programa que la tarea
                        08520   ;       se ha completado.
                        08521   ;
                        08522   ;       El controlador activa también un código de retorno en el bloque de estado
                        08523   ;       de disco -- ambos controladores utilizan para ello la misma posición 0043H.
                        08524   ;       Si el primer byte de este bloque de estado es menor que 80H, significa que
                        08525   ;       ha ocurrido un error. No hay más detalles relevantes respecto a la fijación
                        08526   ;       de estados de este BIOS simple. Nótese que el controlador de disco tiene
                        08527   ;       construidos -- en lógica de varios intentos -- lecturas y escritura que se
                        08528   ;       intentan diez veces antes de que el controlador dé un error.
                        08529   ;
                        08530   ;       La disposición de la tabla de control de disco se muestra a continuación.
                        08531   ;       Nótese que los controladores tienen la capacidad de encadenar tablas de
                        08532   ;       control de manera que pueden lanzarse operaciones de disco en cadena. Esta
                        08533   ;       característica no se usa en este BIOS. El controlador precisa, sin embargo,
                        08534   ;       que los apuntadores de cadena en la tabla de control de disco sean
                        08535   ;       apuntadas por los bytes de control principal para indicar el final de la
                        08536   ;       cadena.
                        08537   ;
0040 =                  08538   Disk$Control$8          EQU     40H     ;Byte de control de 8"
0041 =                  08539   Command$Block$8         EQU     41H     ;Apuntador a tabla de control
                        08540   ;
0043 =                  08541   Disk$Status$Block       EQU     43H     ;Bloque de estado de 8" y de 5 1/4"
                        08542   ;
0045 =                  08543   Disk$Control$5          EQU     45H     ;Byte de control de 5 1/4"
0046 =                  08544   Command$Block$5         EQU     46H     ;Apuntador a tabla de control
                        08545   ;
                        08546   ;
                        08547   ;       Floppy Disk Control Tables
                        08548   ;
0B1D 00                 08549   Floppy$Command:                 DB      0       ;Orden
0001 =                  08550   Floppy$Read$Code                EQU     01H
0002 =                  08551   Floppy$Write$Code               EQU     02H
0B1E 00                 08552   Floppy$Unit:                    DB      0       ;Unidad número 0 ó 1
0B1F 00                 08553   Floppy$Head:                    DB      0       ;Cabeza número 0 ó 1
0B20 00                 08554   Floppy$Track:                   DB      0       ;Número de pista
0B21 00                 08555   Floppy$Sector:                  DB      0       ;Número de sector
0B22 0000               08556   Floppy$Byte$Count:              DW      0       ;Número de bytes a leer/escribir
0B24 0000               08557   Floppy$DMA$Address:             DW      0       ;Dirección de transferencia
0B26 0000               08558   Floppy$Next$Status$Block:       DW      0       ;Apuntador a bloque de estado sig.
                        08559                                                   ;  si existen órdenes encadenadas
0B28 0000               08560   Floppy$Next$Control$Location:   DW      0       ;Apuntador a byte de control sig.
                        08561                                                   ;  si existen órdenes encadenadas
                        08562   ;
                        08700   ;#
                        08701   ;
                        08702   ;
                        08703   Write$No$Deblock:                       ;Escritura del contenido de la memoria intermedia
                        08704                                           ;  en el sector correcto
0B2A 3E02               08705           MVI     A,Floppy$Write$Code     ;Obtención de código de función de escritura
0B2C C3310B             08706           JMP     Common$No$Deblock       ;Ir a programa común
                        08707   Read$No$Deblock:                        ;Lectura previa del sector seleccionado en la
                        08708                                           ;  memoria intermedia
0B2F 3E01               08709           MVI     A,Floppy$Read$Code      ;Obtención código de función de lectura
                        08710   Common$No$Deblock:
0B31 321D0B             08711           STA     Floppy$Command  ;Activación código de función de orden
                        08712                                   ;Activación tabla de órdenes no desbloqueadas bytes
                        08713                                   ;  por sector
0B34 3A360A             08714           LDA     Selected$Disk$Type      ;Comprobación de si operación de disco de
0B37 FE03               08715           CPI     M$Disk                  ;  memoria
0B39 CA7A0B             08716           JZ      M$Disk$Transfer ;Sí, es M$Disk
                        08717
                        08718   No$Deblock$Retry:                       ;Punto de reentrada para intento después de error
0B3C 218000             08719           LXI     H,128           ;Bytes por sector
0B3F 22220B             08720           SHLD    Floppy$Byte$Count
0B42 AF                 08721           XRA     A               ;El disquete de 8" tiene únicamente la cabeza 0
0B43 321F0B             08722           STA     Floppy$Head
                        08723                                   ;
0B46 3A2D0A             08724           LDA     Selected$Disk   ;El controlador de disquetes de 8"  sólo tiene
                        08725                                   ;  información en unidades 0 y 1 de manera que
                        08726                                   ;  Selected$Disk debe convertirse
```

Figura 8-10. (Continuación.)

```
0B49 E601      08727            ANI     01H              ;Transformación a 1 ó 0
0B4B 321E0B    08728            STA     Floppy$Unit      ;Fijar número de unidad
               08729                                     ;
0B4E 3A2E0A    08730            LDA     Selected$Track
0B51 32200B    08731            STA     Floppy$Track     ;Fijar número de pista
               08732                                     ;
0B54 3A300A    08733            LDA     Selected$Sector
0B57 32210B    08734            STA     Floppy$Sector    ;Fijar número de sector
               08735
0B5A 2AA609    08736            LHLD    DMA$Address      ;Transferencia directa entre dirección DMA
0B5D 22240B    08737            SHLD    Floppy$DMA$Address       ; y controlador de 8"
               08738                                     ;
               08739                                     ;El controlador de disco acepta tablas de control de
               08740                                     ;  disco encadenadas, pero en este caso no se
               08741                                     ;  utilizan, de manera que los apuntadores al
               08742                                     ;  "siguiente" deben apuntar a los bytes de control
               08743                                     ;  inicial en la página base
0B60 214300    08744            LXI     H,Disk$Status$Block             ;Apuntar estado siguiente al bloque
0B63 22260B    08745            SHLD    Floppy$Next$Status$Block        ; de estado principal
               08746
0B66 214000    08747            LXI     H,Disk$Control$8                ;Apuntar byte de control del
0B69 22280B    08748            SHLD    Floppy$Next$Control$Location    ;  siguiente al byte de control princ.
               08749                                                    ;
0B6C 211D0B    08750            LXI     H,Floppy$Command        ;Apuntar controlador a tabla de control
0B6F 224100    08751            SHLD    Command$Block$8
               08752                                           ;
0B72 214000    08753            LXI     H,Disk$Control$8        ;Activar controlador para realizar
0B75 3680      08754            MVI     M,80H                   ;  la operación
0B77 C33B0C    08755            JMP     Wait$For$Disk$Complete
               08756
               08757   ;
               08900   ;#
               08901   ;       Controlador de disco de memoria
               08902   ;
               08903   ;       Esta rutina debe utilizar una memoria interna intermedia,
               08904   ;       ya que la dirección DMA del banco ("pista") 0 ocupa el mismo lugar
               08905   ;       en el espacio completo de direcciones que el propio M$Disk.
               08906   ;       El M$Disk$Buffer está sobre la marca de 48K y, por tanto,
               08907   ;       permanece en el espacio de la dirección cualquiera que sea
               08908   ;       el banco/pista seleccionado.
               08909   ;
               08910   ;
               08911   ;       Para escritura, debe ser procesado un sector de 128 bytes:
               08912   ;
               08913   ;               1. Trasladar sector DMA$Address -> M$Disk$Buffer
               08914   ;               2. Selección de pista correcta (+1 para obtener número de banco)
               08915   ;               3. Traslado de sector M$Disk$Buffer -> M$Disk imagen
               08916   ;               4. Seleccionar banco 0
               08917   ;
               08918   ;       Para lectura, el proceso es:
               08919   ;
               08920   ;               1. Selección pista/banco correcto
               08921   ;               2. Trasladar sector M$Disk imagen -> M$Disk$Buffer
               08922   ;               3. Seleccionar banco 0
               08923   ;               4. Trasladar sector M$Disk$Buffer -> DMA$Address
               08924   ;
               08925   ;       Si existe el riesgo de que una interrupción sea la causa de que el control
               08926   ;       se transfiera a una dirección por debajo de 48K, las interrupciones
               08927   ;       deben ser deactivadas cuando se selecciona un banco distinto de 0.
               08928   ;
               08929   M$Disk$Transfer:
0B7A 3A300A    08930            LDA     Selected$Sector ;Calcular dirección en memoria multiplicando
0B7D 6F        08931            MOV     L,A             ;  sector * 128
0B7E 2600      08932            MVI     H,0
0B80 29        08933            DAD     H               ;* 2
0B81 29        08934            DAD     H               ;* 4
0B82 29        08935            DAD     H               ;* 8
0B83 29        08936            DAD     H               ;* 16
0B84 29        08937            DAD     H               ;* 32
0B85 29        08938            DAD     H               ;* 64
0B86 29        08939            DAD     H               ;* 128
               08940
0B87 3A2E0A    08941            LDA     Selected$Track  ;Calcular de qué mitad de sector de banco se trata
               08942                                    ;  utilizando el bit LS de la pista
0B8A 47        08943            MOV     B,A             ;Almacenar copia para uso posterior
0B8B E601      08944            ANI     1               ;Aislar indicador primera/segunda
```

Figura 8-10. (Continuación.)

```
0B8D CA940B   08945            JZ      M$Disk$Lower$Half
              08946
0B90 110060   08947            LXI     D,(48 * 1024) / 2       ;Primera mitad, actuar en dirección
0B93 19       08948            DAD     D
              08949
              08950  M$Disk$Lower$Half:                ;HL -> sector en memoria
0B94 78       08951            MOV     A,B             ;Recuperación pista seleccionada
0B95 1F       08952            RAR                     ;División por dos para 33 número de banco
0B96 3C       08953            INR     A               ;Banco 1 es la primera pista
0B97 47       08954            MOV     B,A             ;Preservar para utilización posterior
              08955
0B98 3A1D0B   08956            LDA     Floppy$Command  ;Comprobación de si se trata de lectura o escritura
0B9B FE02     08957            CPI     Floppy$Write$Code
0B9D CABE0B   08958            JZ      M$Disk$Write    ;Escritura
              08959                                    ;Lectura
              08960
0BA0 CDDD0B   08961            CALL    Select$Bank     ;Seleccionar banco de memoria correcto
0BA3 113023   08962            LXI     D,M$Disk$Buffer ;DE -> M$Disk$Buffer, HL -> M$Disk imagen
0BA6 0E10     08963            MVI     C,128/8         ;Número de bloques de 8 bytes a trasladar
0BA8 CDF80A   08964            CALL    Move$8
              08965
0BAB 0600     08966            MVI     B,0             ;Vuelta a banco de memoria normal
0BAD CDDD0B   08967            CALL    Select$Bank
              08968
0BB0 2AA609   08969            LHLD    DMA$Address     ;Obtención dirección de DMA de usuario
0BB3 113023   08970            LXI     D,M$Disk$Buffer
0BB6 EB       08971            XCHG                    ;DE -> DMA de usuario, HL -> M$Disk memoria intermedia
0BB7 0E10     08972            MVI     C,128/8         ;Número de bloques de 8 bytes a trasladar
0BB9 CDF80A   08973            CALL    Move$8
              08974
0BBC AF       08975            XRA     A               ;Indicar no error
0BBD C9       08976            RET
              08977
              08978  M$Disk$Write:                    ;Escritura
0BBE E5       08979            PUSH    H               ;Almacenamiento de direc. de sec. en la imagen M$Disk
0BBF 2AA609   08980            LHLD    DMA$Address     ;Trasladar sector a M$Disk$Buffer
0BC2 113023   08981            LXI     D,M$Disk$Buffer
0BC5 0E10     08982            MVI     C,128/8         ;Número de bloques de 8 bytes a trasladar
0BC7 CDF80A   08983            CALL    Move$8          ;(No utilizar el registro B)
              08984                                    ;B = banco de memoria a seleccionar
0BCA CDDD0B   08985            CALL    Select$Bank
              08986
0BCD D1       08987            POP     D               ;Recuperación de dirección de imagen de sector M$Disk
0BCE 213023   08988            LXI     H,M$Disk$Buffer
0BD1 0E10     08989            MVI     C,128/8
0BD3 CDF80A   08990            CALL    Move$8          ;Trasladar a imagen M$Disk
              08991
0BD6 0600     08992            MVI     B,0             ;Seleccionar banco 0
0BD8 CDDD0B   08993            CALL    Select$Bank
              08994
0BDB AF       08995            XRA     A               ;Indicar no error
0BDC C9       08996            RET
              08997  ;
              09100  ;*
              09101  ;       Selección de banco
              09102  ;
              09103  ;       Esta rutina conmuta al banco de memoria deseado.
              09104  ;       Nótese que el puerto hardware que controla la selección de bancos
              09105  ;       tiene también otros bits; éstos se preservan a lo largo
              09106  ;       de las selecciones de banco.
              09107  ;
              09108  ;       Parámetros de entrada
              09109  ;
              09110  ;               B = número de banco
              09111  ;
0040 =        09112  Bank$Control$Port        EQU     40H
00F8 =        09113  Bank$Mask                EQU     1111$1000B      ;Para preservar los otros bits
              09114  ;
              09115  Select$Bank:
0BDD DB40     09116            IN      Bank$Control$Port       ;Obtención configuración de puerto en curso
0BDF E6F8     09117            ANI     Bank$Mask               ;Preservar los demás bits
0BE1 B0       09118            ORA     B                       ;Fijación código de banco
0BE2 D340     09119            OUT     Bank$Control$Port       ;Seleccionar banco
0BE4 C9       09120            RET
              09121  ;
              09200  ;*
              09201  ;
```

Figura 8-10. (Continuación.)

```
                09202
                09203   Write$Physical:                 ;Escritura del contenido de la memoria en disco
                09204                                   ;  en el sector correcto
0BE5 3E02       09205           MVI     A,Floppy$Write$Code     ;Obtención código función de escritura
0BE7 C3EC0B     09206           JMP     Common$Physical ;Ir código común
                09207   Read$Physical:                  ;Lectura previa del sector seleccionado
                09208                                   ;  en la memoria en disco
0BEA 3E01       09209           MVI     A,Floppy$Read$Code      ;Obtención de código de función de lectura
                09210   ;
                09211   Common$Physical:
0BEC 321D0B     09212           STA     Floppy$Command  ;Fijación tabla de órdenes
                09213
                09214   ;
                09215   Deblock$Retry:                  ;Punto de reent. para nuevo intento después de error
0BEF 3A2A0A     09216           LDA     In$Buffer$Disk$Type     ;Obtención tipo de disco en curso en memoria
0BF2 FE01       09217           CPI     Floppy$5                ;Confirmación que se trata de disq. de 5 1/4"
0BF4 CAFD0B     09218           JZ      Correct$Disk$Type       ;Sí
0BF7 3E01       09219           MVI     A,1                     ;No, indicación error en disco
0BF9 32320A     09220           STA     Disk$Error$Flag
0BFC C9         09221           RET
                09222   Correct$Disk$Type:              ;Activación tabla de control de disco
                09223                                   ;
0BFD 3A270A     09224           LDA     In$Buffer$Disk  ;Conversión número de disco a 1 ó 0
0C00 E601       09225           ANI     1               ;  para controlador de disco
0C02 321E0B     09226           STA     Floppy$Unit
                09227
0C05 2A280A     09228           LHLD    In$Buffer$Track ;Fijación de número de cabeza y de pista
0C08 7D         09229           MOV     A,L             ;Pistas con número impar estarán en cabeza  0,
0C09 E601       09230           ANI     1               ;  las restantes en cabeza 1
0C0B 321F0B     09231           STA     Floppy$Head     ;Fijación número de cabeza
                09232
0C0E 7D         09233           MOV     A,L             ;Nota: Se trata de un valor de 1 solo byte
0C0F 1F         09234           RAR                     ; /2 para pista (acarreo a 0 desde ANI anterior)
0C10 32200B     09235           STA     Floppy$Track
                09236
0C13 3E01       09237           MVI     A,1                     ;Comienzo con sector 1 de la pista completa
0C15 32210B     09238           STA     Floppy$Sector           ;  a transferir
                09239                                           ;
0C18 210012     09240           LXI     H,Bytes$Per$Track       ;Fijación contador de bytes para pista
0C1B 22220B     09241           SHLD    Floppy$Byte$Count       ;  completa a transferir
                09242                                           ;
0C1E 21A40F     09243           LXI     H,Disk$Buffer           ;Fijación de acción de transferencia para
0C21 22240B     09244           SHLD    Floppy$DMA$Address      ;  memoria de disco
                09245                                           ;
                09246                                           ;Como sólo se utiliza una única tabla
                09247                                           ;  de control, cerrar el estado y dirigir
                09248                                           ;  la cadena de apuntadores a los bytes
                09249                                           ;  de control principal
0C24 214300     09250           LXI     H,Disk$Status$Block
0C27 22260B     09251           SHLD    Floppy$Next$Status$Block
0C2A 214500     09252           LXI     H,Disk$Control$5
0C2D 22280B     09253           SHLD    Floppy$Next$Control$Location
                09254
0C30 211D0B     09255           LXI     H,Floppy$Command        ;Fijación apuntador bloque de comandos
0C33 224600     09256           SHLD    Command$Block$5
                09257
0C36 214500     09258           LXI     H,Disk$Control$5        ;Activación controlador de disco de 5 1/4"
0C39 3680       09259           MVI     M,80H
                09260   ;
                09261   Wait$For$Disk$Complete:         ;Espera hasta que el bloque de estado de disco
                09262                                   ;  indique operación finalizada, entonces
                09263                                   ;  comprobación de ocurrencia de errores
                09264                                   ;En entrada, HL -> bytes de control de disco
0C3B AF         09265           XRA     A               ;asegurar indicador de suspensión a 0
0C3C 32330A     09266           STA     Disk$Hung$Flag
                09267
0C3F 21570C     09268           LXI     H,Disk$Timed$Out        ;Fijación de temporizador centinela
0C42 015802     09269           LXI     B,Disk$Timer            ;Retardo de tiempo
0C45 CD6D08     09270           CALL    Set$Watchdog
                09271   Disk$Wait$Loop:
0C48 7E         09272           MOV     A,M                     ;Obtención byte de control
0C49 B7         09273           ORA     A
0C4A CA5D0C     09274           JZ      Disk$Complete           ;Operación realizada
                09275
0C4D 3A330A     09276           LDA     Disk$Hung$Flag          ;Comprobación de tiempo excedido
0C50 B7         09277           ORA     A
```

Figura 8-10. (Continuación.)

```
0C51 C2B40D    09278            JNZ     Disk$Error              ;Se fijará a 40H
               09279
0C54 C3480C    09280            JMP     Disk$Wait$Loop
               09281
               09282    Disk$Timed$Out:                 ;El control llega a este punto desde la rutina de
               09283                                    ;  temporizador centinela -- esto forma parte
               09284                                    ;  efectivamente de la rutina de servicio de interrup.
0C57 3E40      09285            MVI     A,40H                   ;Fijación código de error de disco suspendido
0C59 32330A    09286            STA     Disk$Hung$Flag          ;  en indicador de error para enviar
               09287                                            ;  control fuera del bucle
0C5C C9        09288            RET                             ;Retorno a rutina de temporizador centinela
               09289
               09290    Disk$Complete:
0C5D 010000    09291            LXI     B,0                     ;Puesta a 0 temporizador centinela
               09292                                            ;HL es aquí irrelevante
0C60 CD6D08    09293            CALL    Set$Watchdog
               09294
0C63 3A4300    09295            LDA     Disk$Status$Block       ;Finalizado, comprobación de estado
0C66 FE80      09296            CPI     80H                     ;Comprobación de si han ocurrido errores
0C68 DAB40D    09297            JC      Disk$Error              ;Si
               09298    ;
               09299    Disk$Error$Ignore:
0C6B AF        09300            XRA     A                       ;No
0C6C 32320A    09301            STA     Disk$Error$Flag         ;Borrado de indicador de error
0C6F C9        09302            RET
               09303
               09304    ;
               09400    ;#
               09401    ;       Manejo de mensajes de error de disco
               09402    ;
               09403    ;
               09404    Disk$Error$Messages:            ;Se examina la tabla comparando el estado de error
               09405                                    ;  del disco y con los de la tabla.
               09406                                    ;  Cuando hay coincidencia o cuando se alcanza
               09407                                    ;  el final de la tabla, la dirección
               09408                                    ;  siguiente al valor del estado apunta
               09409                                    ;  al texto del mensaje correcto.
0C70 40        09410            DB      40H
0C71 9D0C      09411            DW      Disk$Msg$40
0C73 41        09412            DB      41H
0C74 A20C      09413            DW      Disk$Msg$41
0C76 42        09414            DB      42H
0C77 AC0C      09415            DW      Disk$Msg$42
0C79 21        09416            DB      21H
0C7A BC0C      09417            DW      Disk$Msg$21
0C7C 22        09418            DB      22H
0C7D C10C      09419            DW      Disk$Msg$22
0C7F 23        09420            DB      23H
0C80 C80C      09421            DW      Disk$Msg$23
0C82 24        09422            DB      24H
0C83 DA0C      09423            DW      Disk$Msg$24
0C85 25        09424            DB      25H
0C86 E60C      09425            DW      Disk$Msg$25
0C88 11        09426            DB      11H
0C89 F90C      09427            DW      Disk$Msg$11
0C8B 12        09428            DB      12H
0C8C 070D      09429            DW      Disk$Msg$12
0C8E 13        09430            DB      13H
0C8F 140D      09431            DW      Disk$Msg$13
0C91 14        09432            DB      14H
0C92 220D      09433            DW      Disk$Msg$14
0C94 15        09434            DB      15H
0C95 310D      09435            DW      Disk$Msg$15
0C97 16        09436            DB      16H
0C98 3D0D      09437            DW      Disk$Msg$16
0C9A 00        09438            DB      0                       ;<== Terminal
0C9B 4D0D      09439            DW      Disk$Msg$Unknown        ;Código no emparejado
               09440    ;
0003 =         09441    DEM$Entry$Size  EQU     3       ;Tamaño de entrada de la tabla de mensajes de error
               09442    ;                               ;  de disco
               09443    ;       Message texts
               09444    ;
0C9D 48756E670009445    Disk$Msg$40:    DB      'Hung',0        ;Mensaje de tiempo excedido
0CA2 4E6F74205209446    Disk$Msg$41:    DB      'Not Ready',0
0CAC 577269746509447    Disk$Msg$42:    DB      'Write Protected',0
0CBC 446174610009448    Disk$Msg$21:    DB      'Data',0
```

Figura 8-10. (Continuación.)

```
0CC1 466F726D6109449    Disk$Msg$22:    DB      'Format',0
0CC8 4D6973736909450    Disk$Msg$23:    DB      'Missing Data Mark',0
0CDA 427573205409451    Disk$Msg$24:    DB      'Bus Timeout',0
0CE6 436F6E747209452    Disk$Msg$25:    DB      'Controller Timeout',0
0CF9 447269766509453    Disk$Msg$11:    DB      'Drive Address',0
0D07 486561642009454    Disk$Msg$12:    DB      'Head Address',0
0D14 547261636B09455    Disk$Msg$13:    DB      'Track Address',0
0D22 536563746F09456    Disk$Msg$14:    DB      'Sector Address',0
0D31 427573204109457    Disk$Msg$15:    DB      'Bus Address',0
0D3D 496C6C656709458    Disk$Msg$16:    DB      'Illegal Command',0
0D4D 556E6B6E6F09459    Disk$Msg$Unknown:       DB      'Unknown',0
                 09460  ;
                 09461  Disk$EM$1:                      ;Mensajes de error de disco principal -- parte 1
0D55 070D0A      09462                  DB      BELL,CR,LF
0D58 4469736B2009463                    DB      'Disk ',0
                 09464  ;
                 09465                          ;Salida siguiente de texto de error
                 09466  ;
                 09467  Disk$EM$2:                      ;Mensajes de error de disco principal -- parte 2
0D5E 204572726F09468                    DB      ' Error ('
0D66 0000        09469  Disk$EM$Status: DB      0,0     ;Código de estado en hexadecimal
0D68 290D0A202009470                    DB      ')',CR,LF,'     Drive '
0D76 00          09471  Disk$EM$Drive:  DB      0       ;Código de unidad de disco A,B...
0D77 2C2048656109472                    DB      ', Head '
0D7E 00          09473  Disk$EM$Head:   DB      0       ;Número de cabeza
0D7F 2C2054726109474                    DB      ', Track '
0D87 0000        09475  Disk$EM$Track:  DB      0,0     ;Número de pista
0D89 2C2053656309476                    DB      ', Sector '
0D92 0000        09477  Disk$EM$Sector: DB      0,0     ;Número de sector
0D94 2C204F706509478                    DB      ', Operation - '
0DA2 00          09479                  DB      0               ;Terminal
                 09480  ;
0DA3 526561642E09481    Disk$EM$Read:   DB      'Read.',0       ;Nombres de operación
0DA9 577269746509482    Disk$EM$Write:  DB      'Write.',0
                 09483  ;
                 09484  ;
                 09485  Disk$Action$Confirm:
0DB0 00          09486                  DB      0       ;Fijar a carácter entrado por el usuario
0DB1 0D0A00      09487                  DB      CR,LF,0
                 09488  ;
                 09489  ;       ;Procesador de errores de disco
                 09490  ;
                 09491  ;       Esta rutina construye y lanza un mensaje de error.
                 09492  ;       Se plantea entonces al usuario la oportunidad de elegir entre:
                 09493  ;
                 09494  ;               R -- Intentar de nuevo la operación que ha causado el error
                 09495  ;               I -- Ignorar el error e intentar la continuación
                 09496  ;               A -- Abortar el programa y retornar a CP/M
                 09497  ;
                 09498  Disk$Error:
0DB4 F5          09499          PUSH    PSW             ;Preservar código de error desde controlador
0DB5 21660D      09500          LXI     H,Disk$EM$Status        ;Conversión del código del mensaje
0DB8 CD440E      09501          CALL    CAH                     ;Conversión de A a hexadecimal
                 09502
0DBB 3A270A      09503          LDA     In$Buffer$Disk          ;Conversión de identif. de disco para mensaje
0DBE C641        09504          ADI     'A'                     ;Composición de la letra correspondiente
0DC0 32760D      09505          STA     Disk$EM$Drive
                 09506
0DC3 3A1F0B      09507          LDA     Floppy$Head             ;Conversión del número de cabeza
0DC6 C630        09508          ADI     '0'
0DC8 327E0D      09509          STA     Disk$EM$Head
                 09510
0DCB 3A200B      09511          LDA     Floppy$Track            ;Conversión del número de pista
0DCE 21870D      09512          LXI     H,Disk$EM$Track
0DD1 CD440E      09513          CALL    CAH
                 09514
0DD4 3A210B      09515          LDA     Floppy$Sector           ;Conversión del número de sector
0DD7 21920D      09516          LXI     H,Disk$EM$Sector
0DDA CD440E      09517          CALL    CAH
                 09518
0DDD 21550D      09519          LXI     H,Disk$EM$1             ;Salida de la primera parte del mensaje
0DE0 CD5305      09520          CALL    Output$Error$Message
                 09521
0DE3 F1          09522          POP     PSW                     ;Recuperación código de mensaje de error
0DE4 47          09523          MOV     B,A                     ;Para comparaciones
0DE5 216D0C      09524          LXI     H,Disk$Error$Messages - DEM$Entry$Size
```

Figura 8-10. (Continuación.)

```
                09525
                09526           LXI     D,DEM$Entry$Size        ;HL -> tabla -- en entrada
0DE8 110300     09527   Disk$Error$Next$Code:                   ;Obt. de tamaño de la ent. para bucle post.
0DEB 19         09528           DAD     D                       ;Movimiento a la siguiente (o primera) entrada
                09529
0DEC 7E         09530           MOV     A,M                     ;Obtención número de código de la tabla
0DED B7         09531           ORA     A                       ;Comprobación de final de tabla
0DEE CAF80D     09532           JZ      Disk$Error$Matched      ;Sí, coincidencia
0DF1 B8         09533           CMP     B                       ;Comparar a código real
0DF2 CAF80D     09534           JZ      Disk$Error$Matched      ;Sí, salida del bucle
0DF5 C3EB0D     09535           JMP     Disk$Error$Next$Code    ;Comprobación código siguiente
                09536   ;
                09537   Disk$Error$Matched:
0DF8 23         09538           INX     H                       ;HL -> dirección de texto
0DF9 5E         09539           MOV     E,M                     ;Obtención dirección en DE
0DFA 23         09540           INX     H
0DFB 56         09541           MOV     D,M
0DFC EB         09542           XCHG                            ;HL -> texto
0DFD CD5305     09543           CALL    Output$Error$Message    ;Visualización texto explicativo
                09544
0E00 215E0D     09545           LXI     H,Disk$EM$2             ;Visualización segunda parte del mensaje
0E03 CD5305     09546           CALL    Output$Error$Message
                09547
0E06 21A30D     09548           LXI     H,Disk$EM$Read          ;Elección operación de texto
                09549                                           ;  (suposición de lectura)
0E09 3A1D0B     09550           LDA     Floppy$Command          ;Obtención orden de controlador
0E0C FE01       09551           CPI     Floppy$Read$Code
0E0E CA140E     09552           JZ      Disk$Error$Read         ;Sí
0E11 21A90D     09553           LXI     H,Disk$EM$Write         ;No, cambio de dirección en HL
                09554   Disk$Error$Read:
0E14 CD5305     09555           CALL    Output$Error$Message    ;Visualización cambio de operación
                09556   ;
                09557   Disk$Error$Request$Action:              ;Pregunta usuario acción siguiente
0E17 CD2F05     09558           CALL    Request$User$Choice     ;Visualización de indicación y espera para
                09559                                           ;  entrar retorno con A=carácter en mayúscula
0E1A FE52       09560           CPI     'R'                     ;¿Nuevo intento?
0E1C CA2C0E     09561           JZ      Disk$Error$Retry
0E1F FE41       09562           CPI     'A'                     ;Abortar
0E21 CA360E     09563           JZ      System$Reset
0E24 FE49       09564           CPI     'I'                     ;Ignorar
0E26 CA6B0C     09565           JZ      Disk$Error$Ignore
0E29 C3170E     09566           JMP     Disk$Error$Request$Action
                09567   ;
                09568   Disk$Error$Retry:                       ;La decisión de dónde volver depende
                09569                                           ;  de si la operación fallida
                09570                                           ;  ha ocurrido en una unidad desbloqueada
                09571                                           ;  o no desbloqueada
0E2C 3A350A     09572           LDA     Selected$Disk$Deblock
0E2F B7         09573           ORA     A
0E30 C2EF0B     09574           JNZ     Deblock$Retry
0E33 C33C0B     09575           JMP     No$Deblock$Retry
                09576   ;
                09577   System$Reset:                           ;Esto es una decisión radical, pero causa
                09578                                           ;  el reinicio de CP/M
0E36 0E00       09579           MVI     C,0                     ;Reinicialización del sistema
0E38 CD0500     09580           CALL    BDOS
                09581
                09582   ;
                09583   ;
                09584   ;       A a mayúsculas
                09585   ;
                09586   ;       Convierte el contenido del registro A a mayúsculas si está
                09587   ;       en minúsculas.
                09588   ;
                09589   ;       Parámetros de entrada
                09590   ;
                09591   ;               A = carácter a convertir
                09592   ;
                09593   ;       Parámetros de salida
                09594   ;
                09595   ;               A = carácter convertido
                09596   ;
                09597   A$To$Upper:
0E3B FE61       09598           CPI     'a'             ;Comparación con límite inferior
0E3D D8         09599           RC                      ;No hay necesidad de convertir
0E3E FE7B       09600           CPI     'z' + 1         ;Comparación con límite superior
```

Figura 8-10. (Continuación.)

```
0E40 D0          09601            RNC                     ;No hay necesidad de convertir
0E41 E65F        09602            ANI     5FH             ;Conversión a mayúsculas
0E43 C9          09603            RET
                 09604    ;
                 09605    ;       Conversión del registro a hexadecimal
                 09606    ;
                 09607    ;       Esta subrutina convierte el registro A en hexadecimal
                 09608    ;
                 09609    ;       Parámetros de entrada
                 09610    ;
                 09611    ;               A = valor a convertir y lanzar salida
                 09612    ;               HL -> área de memoria intermedia para recibir dos caracteres de salida
                 09613    ;
                 09614    ;       Parámetros de salida
                 09615    ;
                 09616    ;               HL -> byte siguiendo al último byte hexadecimal lanzado
                 09617    ;
                 09618    CAH:
0E44 F5          09619            PUSH    PSW             ;Hacer copia del valor a convertir
0E45 0F          09620            RRC                     ;Desplazamiento de A cuatro lugares a la derecha
0E46 0F          09621            RRC
0E47 0F          09622            RRC
0E48 0F          09623            RRC
0E49 CD4D0E      09624            CALL    CAH$Convert     ;Conversión a ASCII
0E4C F1          09625            POP     PSW             ;Obtención valor original
                 09626                                    ;Salto a rutina que convierte y retorna
                 09627                                    ;  a quien efectúa la llamada
                 09628    CAH$Convert:
0E4D E60F        09629            ANI     0000$1111B      ;Aislamiento de los 4 bits LS
0E4F C630        09630            ADI     '0'             ;Conversión a ASCII
0E51 FE3A        09631            CPI     '9' + 1         ;Comparación con el máximo
0E53 DA580E      09632            JC      CAH$Numeric     ;No hay necesidad de convertir a A -> F
0E56 C607        09633            ADI     7               ;Conversión a letra
                 09634    CAH$Numeric:
0E58 77          09635            MOV     M,A             ;Almacenamiento de carácter
0E59 23          09636            INX     H               ;Actualización de apuntador a carácter
0E5A C9          09637            RET
                 09638
                 09639    ;
                 09640    ;
                 09700    ;#
                 09701    ;
                 09702    ;       Imagen de la tabla de control de disco para arranque caliente
                 09703    ;
                 09704    Boot$Control$Part$1:
0E5B 01          09705            DB      1                       ;Función de lectura
0E5C 00          09706            DB      0                       ;Número de unidad
0E5D 00          09707            DB      0                       ;Número de cabeza
0E5E 00          09708            DB      0                       ;Número de pista
0E5F 02          09709            DB      2                       ;Número de sector de comienzo
0E60 0010        09710            DW      8*512                   ;Número de bytes a leer
0E62 00C4        09711            DW      CCP$Entry               ;Lectura en esta dirección
0E64 4300        09712            DW      Disk$Status$Block       ;Apuntador al bloque de estado siguiente
0E66 4500        09713            DW      Disk$Control$5          ;Apuntador a tabla de control siguiente
                 09714    Boot$Control$Part2:
0E68 01          09715            DB      1                       ;Función de lectura
0E69 00          09716            DB      0                       ;Número de unidad
0E6A 01          09717            DB      1                       ;Número de cabeza
0E6B 00          09718            DB      0                       ;Número de pista
0E6C 01          09719            DB      1                       ;Número de sector de comienzo
0E6D 0006        09720            DW      3*512                   ;Número de bytes a leer
0E6F 00D4        09721            DW      CCP$Entry + (8*512)     ;Lectura en esta dirección
0E71 4300        09722            DW      Disk$Status$Block       ;Apuntador al bloque de estado siguiente
0E73 4500        09723            DW      Disk$Control$5          ;Apuntador a tabla de control siguiente
                 09724
                 09725    ;
                 09726    ;
                 09800    ;#
                 09801    ;
                 09802    WBOOT:          ;Entrada de arranque caliente
                 09803                    ;En arranque caliente, el CCP y el BDOS son cargados
                 09804                    ;  de nuevo en la memoria. En este BIOS
                 09805                    ;  sólo se utilizan disquetes de 5 1/4", con lo que este
                 09806                    ;  programa es específico del hardware del controlador. Se
                 09807                    ;  utilizan dos tablas de control prefabricadas.
0E75 318000      09808            LXI     SP,80H
```

Figura 8-10. (Continuación.)

```
0E78 115B0E    09809            LXI     D,Boot$Control$Part1    ;Ejec. de la primera lec. del arranque caliente
0E7B CD8A0E    09810            CALL    Warm$Boot$Read          ;Cargar unidad 0, pista 0,
               09811                                            ;cabeza 0, sectores 2-8
0E7E 11680E    09812            LXI     D,Boot$Control$Part2    ;Ejecución segunda lectura
0E81 CD8A0E    09813            CALL    Warm$Boot$Read          ;Cargar unidad 0, pista 0,
               09814                                            ;  cabeza 1, sectores 1-3
0E84 CDDF0E    09815            CALL    Patch$CPM               ;Realización de modificaciones personalizadas
0E87 C36C02    09816            JMP     Enter$CPM               ;Fijar página base y entrar en CCP
               09817    ;
               09818    Warm$Boot$Read:                  ;En entrada, DE -> imagen tabla de control
               09819                                    ;Esta tabla de control se transfiere a la tabla
               09820                                    ;  de control del disco principal y
               09821                                    ;  entonces se activa el controlador
0E8A 211D0B    09822            LXI     H,Floppy$Command        ;HL -> tabla de control
0E8D 224600    09823            SHLD    Command$Block$5         ;Indicar al controlador su dirección
               09824                                            ;Transferencia de la imagen de la tabla de
               09825                                            ;  control en la propia tabla de control
0E90 0E0D      09826            MVI     C,13            ;Fijar contador de bytes
               09827    Warm$Boot$Move:
0E92 1A        09828            LDAX    D               ;Obtención byte imagen
0E93 77        09829            MOV     M,A             ;Almacenamiento en tabla de control real
0E94 23        09830            INX     H               ;Actualización de apuntadores
0E95 13        09831            INX     D
0E96 0D        09832            DCR     C               ;Decrementar contador de bytes
0E97 C2920E    09833            JNZ     Warm$Boot$Move  ;Continuar hasta transferencia de todos los bytes
               09834
0E9A 214500    09835            LXI     H,Disk$Control$5        ;Activación controlador
0E9D 3680      09836            MVI     M,80H
               09837    Wait$For$Boot$Complete:
0E9F 7E        09838            MOV     A,M                     ;Obtención byte de estado
0EA0 B7        09839            ORA     A                       ;Comprobación de finalización
0EA1 C29F0E    09840            JNZ     Wait$For$Boot$Complete  ;No
               09841                                            ;Sí, comprobación de errores
0EA4 3A4300    09842            LDA     Disk$Status$Block
0EA7 FE80      09843            CPI     80H
0EA9 DAAD0E    09844            JC      Warm$Boot$Error         ;Sí, ha ocurrido un error
0EAC C9        09845            RET
               09846    ;
               09847    Warm$Boot$Error:
0EAD 21B60E    09848            LXI     H,Warm$Boot$Error$Message
0EB0 CD5F02    09849            CALL    Display$Message
0EB3 C3750E    09850            JMP     WBOOT                   ;Reinicio de arranque caliente
               09851    ;
               09852    Warm$Boot$Error$Message:
0EB6 0D0A57617209853            DB      CR,LF,'Warm Boot Error - retrying...',CR,LF,0
               09854    ;
               09855    ;
               10000    ;#
               10001    ;
               10002    Ghost$Interrupt:        ; El control llagará a este punto sólo en circunstancias
               10003                           ;   excepcionales, tales como que el controlador
               10004                           ;   haya sido programado para suprimir
               10005                           ;   interrupciones no usuales
               10006                           ;
0ED8 F5        10007            PUSH    PSW                     ;Almacenamiento de registros de preinterrup.
0ED9 3E20      10008            MVI     A,IC$EOI                ;Indicación de final de interrupción
0EDB D3D8      10009            OUT     IC$OCW2$Port
0EDD F1        10010            POP     PSW
0EDE C9        10011            RET
               10012    ;
               10013    ;
               10100    ;#
               10101    ;
               10102    ;       Modificaciones a CP/M
               10103    ;
               10104    ;       Esta rutina representa algunas modificaciones al CCP y al BDOS para
               10105    ;       realizar algunas modificaciones personalizadas
               10106    ;
               10107    ;       Ficheros públicos:
               10108    ;               En sistemas con grandes discos duros es extremadamente útil
               10109    ;               la partición del disco utilizando las posibilidades del número de
               10110    ;               usuario; sin embargo, significa un derroche de espacio y disco,
               10111    ;               puesto que deben almacenarse múltiples copias de programas comunes en
               10112    ;               distintas áreas de usuario. Esta modificación hace público al
               10113    ;               usuario 0 -- accesible desde cualquier otra área de usuario.
               10114    ;               *** ATENCION ***
```

Figura 8-10. (Continuación.)

```
                        10115   ;               Los ficheros en el usuario 0 deben fijarse a modo sistema y estado
                        10116   ;               de sólo lectura para prevenir que sean averiados accidentalmente.
                        10117   ;               Debido a los efectos laterales asociados con los ficheros públicos,
                        10118   ;               la modificación puede ser activada o inhibida utilizando
                        10119   ;               un indicador en el bloque de configuración a largo plazo.
                        10120   ;
                        10121   ;       Indicación de usuario:
                        10122   ;               Cuando se utiliza la orden USER de CP/M, y en
                        10123   ;               general números de usuario, es fácil confundirse y
                        10124   ;               olvidar en qué número de usuario se está. Esta
                        10125   ;               modificación hace que el CCP visualice una indicación
                        10126   ;               que muestra no sólo el disco por defecto, sino
                        10127   ;               también el número de usuario en curso y una
                        10128   ;               indicación de si están disponibles los ficheros públicos:
                        10129   ;
                        10130   ;                                       P3B>  o   3B>
                        10131   ;                                       ^
                        10132   ;                                       Cuando los ficheros públicos están disponibles
                        10133   ;
                        10134   ;       Equivalencias para ficheros públicos
                        10135   ;
D35E =                  10136   PF$BDOS$Exit$Point      EQU     BDOS$Entry + 758H
D37C =                  10137   PF$BDOS$Char$Matches    EQU     BDOS$Entry + 776H
D361 =                  10138   PF$BDOS$Resume$Point    EQU     BDOS$Entry + 75BH
000D =                  10139   PF$BDOS$Unused$Bytes    EQU     13
                        10140   ;
                        10141   ;
                        10142   ;       Equivalencias para indicación de usuario
                        10143   ;
C788 =                  10144   UP$CCP$Exit$Point       EQU     CCP$Entry + 388H
C78B =                  10145   UP$CCP$Resume$Point     EQU     CCP$Entry + 38BH
C513 =                  10146   UP$CCP$Get$User         EQU     CCP$Entry + 113H
C5D0 =                  10147   UP$CCP$Get$Disk$Id      EQU     CCP$Entry + 1D0H
C48C =                  10148   UP$CCP$CONOUT           EQU     CCP$Entry + 8CH
                        10149   ;
                        10150   ;
                        10151   ;       Fijación de punto de intervención
                        10152   ;
                        10153   Patch$CPM:
0EDF 3EC3               10154           MVI     A,JMP           ;Fijación de código de operación
0EE1 325ED3             10155           STA     PF$BDOS$Exit$Point
0EE4 3288C7             10156           STA     UP$CCP$Exit$Point
0EE7 21F40E             10157           LXI     H,Public$Patch
0EEA 225FD3             10158           SHLD    PF$BDOS$Exit$Point + 1
0EED 2111OF             10159           LXI     H,Prompt$Patch  ;Obtención de dirección del programa de intervención
0EF0 2289C7             10160           SHLD    UP$CCP$Exit$Point + 1
                        10161
0EF3 C9                 10162           RET                     ;Retorno para entrar CP/M
                        10163   ;
                        10164   ;
                        10165   ;
                        10166   Public$Patch:                   ;El control llega a este punto desde el BDOS
                        10167                                   ;El BDOS está en proceso de examinar el nombre
                        10168                                   ;  de fichero que se desea en la función
                        10169                                   ;  de búsqueda del siguiente
                        10170                                   ;  HL -> nombre del fichero buscado
                        10171                                   ;  DE -> entrada del directorio
                        10172                                   ;  B = contador de caracteres
                        10173
0EF4 3A4200             10174           LDA     CB$Public$Files ;Comprob. de si los ficheros públicos están activados
0EF7 B7                 10175           ORA     A
0EF8 CA0B0F             10176           JZ      No$Public$Files ;No
                        10177
0EFB 78                 10178           MOV     A,B             ;Obtención de cuenta de caracteres
0EFC B7                 10179           ORA     A               ;Comprobación de si se examina el primer byte
                        10180                                   ;  (que contiene el número de usuario)
0EFD C20B0F             10181           JNZ     No$Public$Files ;No, ignorar modificación
                        10182
0F00 1A                 10183           LDAX    D               ;Obtención núm. de usuario desde entrada al directorio
0F01 FEE5               10184           CPI     0E5H            ;Comprobación de entrada al directorio activa
0F03 CA0B0F             10185           JZ      No$Public$Files ;Sí, ignorar modificación
                        10186
0F06 7E                 10187           MOV     A,M             ;Obtención número de usuario
0F07 B7                 10188           ORA     A               ;Comprobación de si usuario 0
0F08 CA7CD3             10189           JZ      PF$BDOS$Char$Matches    ;Forzar emparejamiento de caracteres
                        10190
```

Figura 8-10. (Continuación.)

```
                    10191   No$Public$Files:                ;Programa de modificaciones reemplazado
0F0B 78             10192           MOV     A,B             ;Comprobación de si el contador indica que los
0F0C FE0D           10193           CPI     PF$BDOS$Unused$Bytes    ;  registros apuntan a campos de bytes
                    10194                                   ;  del FCB no utilizados
0F0E C361D3         10195           JMP     PF$BDOS$Resume$Point    ;Retorno al BDOS
                    10196   ;
                    10197   Prompt$Patch:                   ;El control llega a este punto desde el CCP
                    10198                                   ;El CCP acaba de obtener el id. de unidad
                    10199                                   ;  cuando el control llega a este punto
                    10200                                   ;La versión CCP de CONOUT se utiliza,
                    10201                                   ;  puesto que el CCP mantiene información
                    10202                                   ;  de la posición del cursor
                    10203
0F11 3A4200         10204           LDA     CB$Public$Files ;Comprob. de si están activados los ficheros públicos
0F14 B7             10205           ORA     A
0F15 CA1D0F         10206           JZ      UP$Private$Files        ;No
                    10207
0F18 3E50           10208           MVI     A,'P'
0F1A CD8CC4         10209           CALL    UP$CCP$CONOUT   ;Utilización de la rutina CONOUT del CCP
                    10210
                    10211   UP$Private$Files:
0F1D CD13C5         10212           CALL    UP$CCP$Get$User ;Obtención del número de usuario en curso
0F20 FE0A           10213           CPI     9 + 1           ;Comprobación de si uno o dos dígitos
0F22 D2300F         10214           JNC     UP$2$Digits
0F25 C630           10215           ADI     '0'             ;Conversión a ASCII
                    10216   UP$1$Digit:
0F27 CD8CC4         10217           CALL    UP$CCP$CONOUT   ;Salida de carácter
0F2A CDD0C5         10218           CALL    UP$CCP$Get$Disk$Id      ;Obtención de identificador de disco
0F2D C38BC7         10219           JMP     UP$CCP$Resume$Point     ;Retorno al CCP
                    10220   ;
                    10221   UP$2$Digits:
0F30 C626           10222           ADI     '0' - 10        ;Restar 10 y conversión a ASCII
0F32 F5             10223           PUSH    PSW             ;Almacenamiento segundo dígito convertido
0F33 3E31           10224           MVI     A,'1'           ;Salida primer dígito '1'
0F35 CD8CC4         10225           CALL    UP$CCP$CONOUT
0F38 F1             10226           POP     PSW             ;Recuperación de segundo dígito
0F39 C3270F         10227           JMP     UP$1$Digit      ;Salida del resto de indicación y retorno
                    10228                                   ;  al CCP
                    10229
                    10230   ;
                    10300   ;*
                    10301   ;
                    10302   ;       Dirección de obtención del bloque de configuración
                    10303   ;
                    10304   ;       Esta rutina es llamada por los programas de utilidad que se están
                    10305   ;       ejecutando en la TPA. Dado un número de código específico, devuelve la
                    10306   ;       dirección de un objeto específico en el bloque de configuración.
                    10307   ;
                    10308   ;       Utilizando esta rutina, los programas de utilidad no necesitan conocer
                    10309   ;       exactamente la distribución del bloque de configuración.
                    10310   ;
                    10311   ;       Parámetros de entrada
                    10312   ;
                    10313   ;               C = código identificador del objeto (en efecto, esto es
                    10314   ;                       el subíndice de la dirección del objeto
                    10315   ;                       en la tabla que sigue)
                    10316   ;
                    10317   ;=========================
                    10318   CB$Get$Address:                         ;<=== Punto de entrada al BDOS (privado)
                    10319   ;=========================
0F3C F5             10320           PUSH    PSW                     ;Almacenamiento registros de usuario
0F3D C5             10321           PUSH    B
0F3E D5             10322           PUSH    D
                    10323
0F3F 69             10324           MOV     L,C                     ;Composición en una palabra
0F40 2600           10325           MVI     H,0
0F42 29             10326           DAD     H                       ;Conversión del código en desplaz. de palabra
0F43 114F0F         10327           LXI     D,CB$Object$Table       ;Obtención dirección base de tabla
0F46 19             10328           DAD     D                       ;HL -> dirección del objeto en la tabla
0F47 5E             10329           MOV     E,M                     ;Obtención byte LS
0F48 23             10330           INX     H
0F49 56             10331           MOV     D,M                     ;Obtención byte MS
0F4A EB             10332           XCHG                            ;HL = dirección del objeto
                    10333
0F4B D1             10334           POP     D                       ;Recuperación de los registros de usuario
0F4C C1             10335           POP     B
```

Figura 8-10. (Continuación.)

```
0F4D F1         10336           POP     PSW
                10337
0F4E C9         10338           RET
                10339   ;
                10400   ;#
                10401   ;
                10402   CB$Object$Table:
                10403                                   ;       Código
                10404                                   ;        vv
0F4F 8F0F       10405           DW      Date                    ;01 fecha en ASCII
0F51 990F       10406           DW      Time$In$ASCII           ;02 hora en ASCII
0F53 A30F       10407           DW      Time$Date$Flags         ;03 indicadores de hora, fecha activos
0F55 8D0F       10408           DW      CB$Forced$Input         ;04 apuntador de entrada forzada
0F57 4300       10409           DW      CB$Startup              ;05 mensaje de inicio del sistema
                10410                                           ;   palabra de redireccionamiento
0F59 5800       10411           DW      CB$Console$Input        ;06
0F5B 5A00       10412           DW      CB$Console$Output       ;07
0F5D 5C00       10413           DW      CB$Auxiliary$Input      ;08
0F5F 5E00       10414           DW      CB$Auxiliary$Output     ;09
0F61 6000       10415           DW      CB$List$Input           ;10
0F63 6200       10416           DW      CB$List$Output          ;11
                10417
0F65 6400       10418           DW      CB$Device$Table$Addresses ;12
0F67 B500       10419           DW      CB$12$24$Clock          ;13 Selec. del formato de reloj 12/24 horas
0F69 BD00       10420           DW      RTC$Ticks$per$Second    ;14
0F6B BF00       10421           DW      RTC$Watchdog$Count      ;15
0F6D C100       10422           DW      RTC$Watchdog$Address    ;16
0F6F C300       10423           DW      CB$Function$Key$Table   ;17
0F71 1B02       10424           DW      CONOUT$Escape$Table     ;18
                10425
0F73 8400       10426           DW      D0$Initialize$Stream    ;19
0F75 9100       10427           DW      D0$Baud$Rate$Constant   ;20
0F77 9400       10428           DW      D1$Initialize$Stream    ;21
0F79 A100       10429           DW      D1$Baud$Rate$Constant   ;22
0F7B A400       10430           DW      D2$Initialize$Stream    ;23
0F7D B100       10431           DW      D2$Baud$Rate$Constant   ;24
0F7F 4002       10432           DW      Interrupt$Vector        ;25
0F81 890F       10433           DW      LTCB$Offset             ;26
0F83 8B0F       10434           DW      LTCB$Length             ;27
0F85 4200       10435           DW      CB$Public$Files         ;30
0F87 A421       10436           DW      Multi$Command$Buffer    ;31
                10437   ;
                10500   ;#
                10501   ;       Bloque de configuración a corto plazo.
                10502   ;
                10503   ;       Contiene variables que serán fijadas una vez
                10504   ;       iniciado CP/M, pero que no serán perservadas de una carga
                10505   ;       de CP/M en la siguiente; esta parte del bloque
                10506   ;       de configuración forma los últimos bytes inicializados
                10507   ;       en el BIOS.
                10508   ;
                10509   ;       Los dos valores posteriores se utilizan por los programas de utilidad
                10510   ;       que necesitan leer en el bloque de configuración a largo plazo
                10511   ;       del disco. El BIOS comienza al principio de una página de 256 bytes
                10512   ;       y, sin embargo, estará siempre al comienzo de un sector de 128 bytes
                10513   ;       en el área reservada del disco. Un programa de utilidad puede
                10514   ;       entonces, utilizando la llamada privada al BIOS, determinar cuántos
                10515   ;       sectores de 128 bytes es preciso leer a partir de la fórmula:
                10516   ;
                10517   ;               (LCTB$Offset + LTCB$Length) / 128
                10518   ;
                10519   ;       El LTCB$Offset es el desplazamiento desde el comienzo BIOS,
                10520   ;       donde comienza el primer byte del bloque de configuración
                10521   ;       a largo plazo. Utilizando el desplazamiento y la longitud, la utilidad
                10522   ;       puede copiar la versión RAM del LTCB sobre la imagen del disco
                10523   ;       que ha leído y escribir entonces el LTCB actualizado
                10524   ;       en disco.
                10525   ;
0F89 BED9       10526   LTCB$Offset:    DW      BIOS$Entry - Long$Term$CB
0F8B E601       10527   LTCB$Length:    DW      Long$Term$CB$End - Long$Term$CB
                10528   ;
                10529   ;       Apuntador de entrada forzada
                10530   ;
                10531   ;       Si CONIN encuentra que este apuntador apunta a un byte no nulo,
                10532   ;       este byte será inyectado en el flujo de entrada de consola
                10533   ;       como si hubiera sido mecanografiado en ella. El apuntador
```

Figura 8-10. (Continuación.)

```
                    10534   ;       es desplazado entonces al siguiente byte de la memoria.
                    10535
0F8D 4300           10536   CB$Forced$Input:        DW      CB$Startup
                    10537   ;
                    10538   ;
                    10539   Date:                   ;Fecha en curso
0F8F 31302F313710540            DB      '10/17/82',LF   ;A menos que se diga lo contrario, esta es la
                    10541                           ;  fecha del sistema actualizada
                    10542                           ;Normalmente será fijada por la utilidad DATE
0F98 00             10543           DB      0       ;Byte terminal 00
                    10544
                    10545   Time$in$ASCII:          ;Hora en curso
0F99 3030           10546   HH:     DB      '00'            ;Horas
0F9B 3A             10547           DB      ':'
0F9C 3030           10548   MM:     DB      '00'            ;Minutos
0F9E 3A             10549           DB      ':'
0F9F 3030           10550   SS:     DB      '00'            ;Segundos
                    10551   Time$in$ASCII$End:              ;Utilizado para actualizar la hora
0FA1 0A             10552           DB      LF
0FA2 00             10553           DB      0               ;Byte terminal 00
                    10554   ;
                    10555   ;
                    10556   Time$Date$Flags:        ;Este byte contiene dos indicadores que se utilizan
                    10557                           ;  para indicar cuándo la hora y/o la fecha
                    10558                           ;  han sido fijadas por programa o utilizando las utilidades
                    10559                           ;  TIME y DATE. Estos indicadores son consultados
                    10560                           ;  por programas de utilidad que necesitan conocer
                    10561                           ;  la fecha y la hora en curso.
0FA3 00             10562           DB      0
0001 =              10563   Time$Set        EQU     0000$0001B
0002 =              10564   Date$Set        EQU     0000$0010B
                    10565
                    10566   ;
                    10700   ;*
                    10701   ;       Areas de memoria intermedia no inicializadas
                    10702   ;
                    10703   ;       Con excepción de Disk$Buffer, que contiene unos pocos bytes de código,
                    10704   ;       el resto de variables no inicializadas está en esa área.
                    10705   ;       Lo que tiene como efecto reducir el número de bytes necesarios
                    10706   ;       para almacenar, en disco, la imagen de CP/M ya que las áreas
                    10707   ;       no inicializadas no necesitan mantenerse en el mismo.
                    10708   ;
                    10709   ;
                    10800   ;*
                    10801   ;
                    10802   ;       El programa de inicialización de arranque en frío es necesario únicamente
                    10803   ;       una vez. Puede ser reescrito destructivamente una vez que ha sido ejecutado.
                    10804   ;       Sin embargo, está "escondido" en la memoria intermedia principal del disco.
                    10805   ;
                    10806   ;
0FA4                10807   Disk$buffer:    DS      Physical$Sector$Size * Physical$Sec$Per$Track
                    10808   ;
                    10809                                   ;Almacenamiento contador de posiciones
21A4 =              10810   After$Disk$Buffer       EQU     $       ;$ = valor en curso del cont. de posiciones
                    10811   ;
0FA4                10812                           ORG     Disk$Buffer ;Puesta a cero del contador de posiciones
                    10813   ;
                    10814   Initialize$Stream:      ;Este flujo de datos se utiliza por la
                    10815                           ;  subrutina de inicialización. Tiene el
                    10816                           ;  siguiente formato:
                    10817                           ;
                    10818                           ;       DB      Número de puerto a inicializar
                    10819                           ;       DB      Número de byte a lanzar a salida
                    10820                           ;       DB      xx,xx,xx,xx datos a lanzar a salida
                    10821                           ;       :
                    10822                           ;       :
                    10823                           ;       DB      Número de puerto terminal 00H
                    10824                           ;
                    10825                           ;
                    10826   ;
                    10827   ;       Flujo de inicialización declarado aquí
0FA4 DB             10828           DB      IC$ICW1$Port    ;Programa el controlador de interrupciones 8259
0FA5 01             10829           DB      1
0FA6 56             10830           DB      IC$ICW1
                    10831
0FA7 D9             10832           DB      IC$ICW2$Port
```

Figura 8-10. (Continuación.)

```
0FA8 01          10833            DB      1
0FA9 02          10834            DB      IC$ICW2
                 10835
0FAA D9          10836            DB      IC$OCW1$Port
0FAB 01          10837            DB      1
0FAC FC          10838            DB      IC$OCW1
                 10839
0FAD 83          10840            DB      83H                     ;Programa el generador de reloj 8253
0FAE 01          10841            DB      1
0FAF 34          10842            DB      00$11$010$0B            ;Contador 0, interrupción periódica, modo 2
                 10843
0FB0 80          10844            DB      80H                     ;RTC utiliza canal 0
0FB1 02          10845            DB      2
0FB2 0146        10846            DW      17921                   ;19721 * 930 nanosegundos =
                 10847                                            ;  16,666 milisegundos). 60 pulsos/seg.
0FB4 00          10848            DB      0               ;Número de puerto terminal 0
                 10849    ;
                 10850    ;
                 10851    Signon$Message:
0FB5 43502F4D2010852              DB      'CP/M 2.2.'
0FBE 3030        10853            DW      VERSION         ;Número de versión en curso
0FC0 20          10854            DB      ' '
0FC1 3032        10855            DW      MONTH           ;Fecha actual
0FC3 2F          10856            DB      '/'
0FC4 3236        10857            DW      DAY
0FC6 2F          10858            DB      '/'
0FC7 3833        10859            DW      YEAR
0FC9 0D0A0A      10860            DB      CR,LF,LF
0FCC 456E68616E10861              DB      'Enhanced BIOS',CR,LF,LF
0FDC 4469736B2010862              DB      'Disk Configuration :',CR,LF,LF
0FF3 202020202010863              DB      '      A: 0.35 Mbyte 5" Floppy',CR,LF
1011 202020202010864              DB      '      B: 0.35 Mbyte 5" Floppy',CR,LF,LF
1030 202020202010865              DB      '      C: 0.24 Mbyte 8" Floppy',CR,LF
104E 202020202010866              DB      '      D: 0.24 Mbyte 8" Floppy',CR,LF
106C 202020202010867              DB      '      M: 0.19 Mbyte Memory Disk',CR,LF,LF
                 10868    ;
108D 00          10869            DB      0
                 10870    ;
                 10871    ;       Mensaje para M$Disk
                 10872    ;
                 10873    M$Disk$Setup$Message:
108E 202020202010874              DB      '      M$Disk already contains valid information.',CR,LF,0
                 10875    M$Disk$Not$Setup$Message:
10C0 202020202010876              DB      '      M$Disk has been initialized to empty state.',CR,LF,0
                 10877    ;
                 10878    M$Disk$Dir$Entry:               ;Entrada ficticia al directorio utilizada para
                 10879                                    ;  determinar si M$Disk contiene información válida
10F3 0F          10880            DB      15              ;Usuario 15
10F4 4D2444697310881              DB      'M$Disk  '
10FC A0A020      10882            DB      ' '+80H,' '+80H,' '     ;Modo sistema y sólo lectura
10FF 00000000    10883            DB      0,0,0,0
1103 000000000010884              DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                 10885    ;
0004 =           10886    Default$Disk    EQU     0004H   ;Disco por defecto en página base
                 10887    ;
                 10888    BOOT:           ;Entrado directamente desde el vector JMP del BIOS
                 10889                    ;El control será transferido a este punto por el
                 10890                    ;  cargador de CP/M
                 10891                    ;
                 10892                            ;Inicialización del sistema
                 10893                            ;Esta rutina utiliza Initialize$Stream
                 10894                            ;  declarada anteriormente
                 10895
1113 F3          10896            DI              ;Desactivación de interrupciones para prevenir
                 10897                            ; efectos laterales durante la inicialización
1114 21A40F      10898            LXI     H,Initialize$Stream     ;HL -> flujo de datos
1117 CD1903      10899            CALL    Output$Byte$Stream      ;Salida a los puertos
                 10900                                            ;  especificados
                 10901
111A CDEE02      10902            CALL    General$CIO$Initialization   ;Inicialización de los dispositivos de
                 10903                                                 ;  carácter
111D 21B50F      10904            LXI     H,Signon$Message        ;Visualización de mensaje de funcionando en
1120 CD5F02      10905            CALL    Display$Message         ;  la consola
                 10906    ;
1123 CDDF0E      10907            CALL    Patch$CPM       ;Realización de las modificaciones necesarias al CCP
                 10908                                    ;  y BDOS para mejoras particulares
```

Figura 8-10. (Continuación.)

```
                    10909
                    10910                                   ;Inicialización de M$Disk
                    10911                                   ;Si el directorio M$Disk contiene el nombre
                    10912                                   ;  especial reservado M$Disk (con minúsculas
                    10913                                   ;  y fijado a modo sistema y sólo lectura),
                    10914                                   ;  se supone que M$Disk contiene datos
                    10915                                   ;  válidos
                    10916                                   ;Si el fichero M$Disk no figura, la entrada
                    10917                                   ;  al directorio M$Disk es transferida
                    10918                                   ;  a la imagen de M$Disk y el resto del
                    10919                                   ;  directorio puesto a 0E5H
1126 0601           10920           MVI     B,1             ;Selección del banco 1
1128 CDDD0B         10921           CALL    Select$Bank     ;  que contiene el directorio M$Disk
                    10922
                    10923                                   ;Comprobación de presencia de entrada M$Disk
112B 210000         10924           LXI     H,0             ;Dirección de comienzo para el primer directorio
112E 11F310         10925           LXI     D,M$Disk$Dir$Entry
1131 0E20           10926           MVI     C,32            ;Longitud a comparar
                    10927   M$Disk$Test:
1133 1A             10928           LDAX    D               ;Obtención de byte de variable inicializada
1134 BE             10929           CMP     M               ;Comparación con imagen M$Disk
1135 C24F11         10930           JNZ     M$Disk$Not$Setup        ;No coincidencia
1138 13             10931           INX     D
1139 23             10932           INX     H
113A 0D             10933           DCR     C
113B CA4111         10934           JZ      M$Disk$Setup    ;Comparados todos los bytes
113E C33311         10935           JMP     M$Disk$Test
                    10936   ;
                    10937   M$Disk$Setup:
1141 218E10         10938           LXI     H,M$Disk$Setup$Message  ;Información al usuario
                    10939   ;
                    10940   M$Disk$Setup$Done:
1144 CD5F02         10941           CALL    Display$Message
                    10942
1147 AF             10943           XRA     A               ;Fijación de unidad de disco por defecto a A:
1148 320400         10944           STA     Default$Disk
114B FB             10945           EI                      ;Pueden activarse ahora las interrupciones
                    10946
114C C36C02         10947           JMP     Enter$CPM       ;Ir a CP/M
                    10948   ;
                    10949   M$Disk$Not$Setup:
114F 110000         10950           LXI     D,0             ;Transferir a la entrada del directorio M$Disk
1152 21F310         10951           LXI     H,M$Disk$Dir$Entry      ;  a la imagen M$Disk
1155 0E04           10952           MVI     C,32/8          ;Número de bloques de 8 bytes a transferir
1157 CDF80A         10953           CALL    Move$8
                    10954   ;
                    10955                                   ;DE -> byte que sigue a la entrada M$Disk
                    10956                                   ;  en la imagen
115A 3EE5           10957           MVI     A,0E5H          ;Activación rellenado de memoria
115C 12             10958           STAX    D               ;Almac. del primer byte en área "fuente"
115D 62             10959           MOV     H,D             ;Fijación de HL a DE +1
115E 6B             10960           MOV     L,E
115F 23             10961           INX     H
1160 0EFC           10962           MVI     C,((2 * 1024) - 32) / 8 ;Dos bloques menores de 32 bytes
                    10963                                   ;  para entrada M$Disk
1162 CDF80A         10964           CALL    Move$8          ;Utilización de Move$8 para oper. de rellenado
                    10965
1165 21C010         10966           LXI     H,M$Disk$Not$Setup$Message
1168 C34411         10967           JMP     M$Disk$Setup$Done       ;Salida de mensaje y entrada en CP/M
                    10968   ;
                    10969   ;
116B 00             10970           DB      0               ;Ficticio
                    10971   Last$Initialized$Byte:          ;<== dirección último byte inicializado
                    10972   ;
                    10973   ;       Final de programa de inicialización de arranque en frío
                    10974   ;
21A4                10975           ORG     After$Disk$Buffer       ;Puesta a cero controlador de posiciones
                    10976   ;
21A4                10977   Multi$Command$Buffer:   DS      128     ;Puede utilizarse para insertar secuencias
                    10978                                           ;  largas de órdenes en el flujo de entrada
                    10979                                           ;  de consola fijando a este punto el
                    10980                                           ;  apuntador de entrada forzada
                    10981   ;
0020 =              10982   DO$Buffer$Length        EQU     32      ;Debe ser un número binario
2224                10983   DO$Buffer:      DS      DO$Buffer$Length
                    10984   ;
```

Figura 8-10. (Continuación.)

```
0020 =          10985   D1$Buffer$Length        EQU     32      ;Debe ser un número binario
2244            10986   D1$Buffer:      DS      D1$Buffer$Length
                10987   ;
0020 =          10988   D2$Buffer$Length        EQU     32      ;Debe ser un número binario
2264            10989   D2$Buffer:      DS      D2$Buffer$Length
                10990   ;
                10991   ;       Areas de datos para control de dispositivos de caracteres
                10992   ;
2284            10993   PI$User$Stack:  DS      2       ;Area de almacenamiento para apuntador a pila
                10994                                   ;  de usuario cuando ocurre una interrupción
2286            10995   PI$User$HL:     DS      2       ;Area de almacenamiento de HL de usuario
2288            10996                   DS      40      ;Area de pila para uso de las rutinas de servicio
                10997   PI$Stack:                       ;  de interrupciones para prevenir desbordamiento
                10998                                   ;  del área de pila de usuario
                10999   ;
22B0            11000   Directory$Buffer:       DS      128     ;Memoria intermedia de directorio de disco
                11001   ;
2330            11002   M$Disk$Buffer:          DS      128     ;Memoria intermedia para
                11003                                           ;  M$Disk
                11004   ;
                11005   ;       Areas de trabajo de disco
                11006   ;
                11007   ;       El BDOS la utiliza para detectar cambios inesperados
                11008   ;       de los disquetes. El BDOS fijará automáticamente
                11009   ;       el disquete cambiado a estado de sólo lectura.
                11010   ;
23B0            11011   Disk$A$Workarea:        DS      32      ; A:
23D0            11012   Disk$B$Workarea:        DS      32      ; B:
23F0            11013   Disk$C$Workarea:        DS      16      ; C:
2400            11014   Disk$D$Workarea:        DS      16      ; D:
                11015   ;
                11016   ;
                11017   ;       Vectores de asignación de disco
                11018   ;
                11019   ;       El BDOS los utiliza para mantener un mapa de bits
                11020   ;       de cuáles son los bloques utilizados y cuáles están libres.
                11021   ;       Utiliza un byte por cada ocho bloques, de ahí
                11022   ;       la expresión de la forma (bloques/8+1).
                11023   ;
2410            11024   Disk$A$Allocation$Vector        DS      (174/8)+1       ; A:
2426            11025   Disk$B$Allocation$Vector        DS      (174/8)+1       ; B:
                11026   ;
243C            11027   Disk$C$Allocation$Vector        DS      (242/8)+1       ; C:
245B            11028   Disk$D$Allocation$Vector        DS      (242/8)+1       ; D:
                11029   ;
247A            11030   M$Disk$Allocation$Vector        DS      (192/8)+1       ; M$Disk
                11031
2493            11032           END     ;Final de listado del BIOS mejorado
```

Figura 8-10. (Continuación.)

Clases de errores
Funciones de manejo de errores del BIOS
Manejo práctico de errores
Errores de entrada/salida de caracteres
Errores de disco
Mejora de mensajes de error

9

Procesamiento de errores de hardware

Este capítulo describe las mejoras que pueden hacerse para mejorar de alguna forma el manejo primitivo de errores del CP/M. Cubre las clases de errores corrientes que el BIOS puede tener que manejar. Describe algunos de los aspectos filosóficos en los que se originan los errores, cómo detectar éstos y cómo corregirlos o cómo sacar el máximo partido de la situación.

Al final del capítulo se presentan algunas subrutinas de ejemplo de procesamiento de errores. Algunas de éstas se han mostrado ya en el capítulo anterior como parte del BIOS mejorado (figura 8-10); aquí se han repetido para que se puedan estudiar aisladamente.

Clases de errores

Básicamente, el usuario percibe sólo dos clases de errores —los que son corregibles por el usuario y los que no lo son—. Existe una tercera clase,

casi invisible, de errores —aquellos que son recuperables por el hardware o el software sin la intervención del usuario.

Las fuentes posibles de error de hardware varían ampliamente de una computadora a otra, ya que la detección de errores depende en gran manera de la lógica particular del hardware. El BIOS puede detectar algunos errores relacionados con el hardware —principalmente errores provocados cuando algo tarda demasiado en ocurrir, como, por ejemplo, cuando una impresora recalcitrante no reacciona en un tiempo específico.

El BDOS no tiene incorporado ningún programa de detección de hardware. Puede detectar errores *de sistema,* tales como un intento de escribir en un fichero de disco que esté marcado "sólo lectura" en el directorio de ficheros o intentar acceder a ficheros que no se encuentran en el disco. Estos errores detectados por el BDOS, sin embargo, generalmente no están relacionados con el estado satisfactorio del hardware. Por ejemplo, un controlador de disco con un problema de hardware podría escribir fácilmente sobre un sector del directorio, borrando con ello algún fichero. Este error no se notaría hasta que el usuario intentase utilizar uno de los ficheros estropeados.

Funciones de manejo de errores del BIOS

El programa de manejo de errores del BIOS deberá servir para las siguientes funciones:

- Detección.
- Análisis.
- Indicación.
- Corrección.

Detección del error

Obviamente, antes de tomar las medidas indicadas, hay que detectar el error. Esto puede hacerse sólo con el software o mediante la interacción del BIOS con lógica de detección de errores en el hardware. En general, los únicos errores que puede detectar el BIOS sin asistencia están causados cuando ciertas operaciones tardan en completarse más de lo que se esperaba. Puesto que el autor del BIOS conoce la forma de operación de los elementos periféricos específicos del sistema, el programa puede predecir cuánto tardará en llevarse a cabo una operación particular y puede señalar

un error cuando el tiempo excede al previsto. Esto incluiría el problema de que las impresoras no reaccionen dentro del período de tiempo especificado.

El BIOS puede trabajar en colaboración con el hardware para determinar si éste ha detectado un error. Armado con las especificaciones del hardware, el BIOS puede introducir información en el controlador o en el estado del dispositivo para disparar la lógica de detección de errores. Cómo se realiza esto depende en gran manera de los periféricos de cada computadora y del grado en que estos dispositivos han "activado" a los controladores para que sean capaces de procesar independientemente de la computadora. Desgraciadamente, muchos fabricantes documentan el significado de los bits de estado individual que indican errores, pero no combinaciones de errores, o la forma de actuar cuando ocurre un error particular.

Análisis de errores

Suponiendo que el BIOS ha detectado un error, lo primero que tiene que hacer es determinar la clase de éste; es decir, si el error puede ser corregido o no simplemente intentando de nuevo la operación. Algunos errores parecen ser corregibles, pero al reintentar la operación varias veces siempre queda inacabada. Un ejemplo sería el error de control de suma (check-sum) al leer un sector del disco. Si se hacen varios intentos para leer el sector y todos terminan en error, se convierte en error "fatal". El programa del BIOS ha de ser capaz de realizar una clasificación inicial y luego de una clasificación subsiguiente si falla la acción llevada a cabo para corregir el error.

Otros tipos de errores pueden ser clasificados inmediatamente como errores fatales —no puede hacerse nada para salvar la situación—. Por ejemplo, si el controlador de disco indica que no puede encontrar un número de sector particular en un disquete (debido a un error en el formato), el BIOS no puede hacer nada más que informar al usuario del problema y suministrarle otra información que pueda servirle de ayuda.

El análisis de errores puede requerir algunas búsquedas básicas, tales como introducción de fallos en el hardware y observación de las combinaciones de indicadores de error. Por ejemplo, algunas impresoras (con una interfase paralela) indican que están "fuera del papel" u "ocupadas" cuando, de hecho, están desconectadas. El BIOS debería detectar esta condición y comunicar al usuario que debe conectar la impresora y no cargar más papel.

Indicación de errores

Un mensaje de error incompleto o críptico es enloquecedor. Es el equivalente funcional de la frase "se ha cometido un error. ¡A ver si aciertas lo que está mal!"

Un mensaje de error, para ser completo, debería informar al que lo recibe de lo siguiente:

- El hecho de que ha ocurrido un error.
- Si se ha intentado o no la recuperación automática y si ésta ha fallado.
- Los detalles del error, si es necesario en términos técnicos para que sirvan a un ingeniero de hardware.
- Cuáles son las posibles soluciones que tiene el usuario.

Para enfocar estos puntos, hay que considerar el mensaje de error que puede ser enviado por el CP/M después de que se haya intentado cargar un programa introduciendo su nombre en el CCP. Lo que aparece en la consola es el siguiente diálogo:

```
A>myprog<cr>
BAD LOAD
A>
```

Lo único que se sabe es que hay un error, y hay que adivinar cuál es, aun cuando la causa específica del error fuera conocida para el CP/M cuando envió el mensaje. Este mensaje de error es enviado por el CCP cuando intenta cargar un fichero ".COM" mayor que el área de programa transitorio actual. El mensaje "BAD LOAD" es únicamente inteligible *después* de saber cuál es el error. Aun entonces, no explica qué es lo equivocado, si se puede hacer algo para solucionarlo y cómo se ha de hacer.

Para ser completo, este mensaje de error podría decir algo así como:

```
A>myprog<cr>
    "MYPROG.COM" exceeds the available memory space by
    1,024 bytes, and therefore cannot be loaded under the
    current version of CP/M.
```

Obsérvese la forma en que el mensaje indica cuál es el problema y lo valora, de forma que se pueda decidir su gravedad (se necesita obtener 1K más de memoria o reducir el tamaño del programa). También indica cómo se encuentra —no se puede cargar este programa con la versión del CP/M actual; por tanto, es inútil reintentar la operación.

A muchos programadores de sistemas no les agrada que surjan mensajes como el del ejemplo. Arguyen que tales mensajes son demasiado largos y trabajosos para algo que no ocurre muy a menudo. Admitido que el mensaje sea demasiado largo, podría reducirse para leerlo en la forma siguiente:

```
(131) Program 1,024 bytes too large to load.
```

Este lleva la misma información; el número entre paréntesis sirve como referencia para un manual en el que haya que describir el impacto total del mensaje.

El problema mayor con la forma en que se diseñan los mensajes de error es que usualmente se escriben por programadores para ser leídos por usuarios que no poseen conocimientos técnicos, y los programadores, como es sabido, no tienen ni idea de lo que necesitan saber los no expertos.

Las indicaciones de error que se diseñen han de poseer las siguientes indicaciones, desde el punto de vista del usuario:

- La causa del error.
- La gravedad del error.
- La acción correctora que se ha llevado a cabo o que se ha de emprender.

Examínense los mensajes de error en el procesador de errores para el BIOS ejemplificado en la figura 8-10, desde la línea 03600 en adelante. Aunque éstos son una mejora del BDOS de propósito general

```
BDOS Error on A: Bad Sector
```

tampoco este mensaje responde realmente a todos los requerimientos de un buen sistema de mensajes de error.

Otro aspecto de los errores que se descuida a menudo es que la mayor parte de los errores de hardware forman un modelo. Este modelo sólo es discernible normalmente al ojo entrenado de un ingeniero de mantenimiento de hardware. Cuando estos ingenieros se consultan para investigar un problema, harán preguntas al usuario para determinar si un fallo dado es un incidente aislado o parte de un modelo repetido. Esta es la razón por la que un mensaje de error debería contener detalles técnicos adicionales. Por ejemplo, un mensaje de error de disco debería incluir la pista y el sector utilizados en la operación que acabó en error. Sólo con estos detalles puede el ingeniero reconstruir el contexto de un fallo o un grupo de fallos.

Corrección de errores

Suponiendo que en la consola haya aparecido un mensaje de error claro, el usuario se enfrenta aun con la cuestión: "Y ahora, ¿qué hago?". Esto no sólo puede ser difícil de contestar para el usuario, sino que incluso la decisión que se haya tomado para una solución particular puede ser difícil de ejecutar para el BIOS.

Normalmente existen tres posibles opciones en respuesta a errores:

- Intentar de nuevo la operación.
- Ignorar el error e intentar continuar.
- Abortar el programa causante del error y volver al CP/M.

Para algunos errores, el reintento puede ser efectivo. Por ejemplo, si se olvida poner la línea de conexión de la impresora y se obtiene un mensaje de error de "impresora fuera de tiempo", es fácil poner la impresora en línea y pedir al BIOS que intente de nuevo enviar los datos a la impresora.

Pocas veces se puede ignorar un error y esperar obtener resultados apreciables de la máquina; muchos controladores de disco no transfieren tampoco datos entre sí mismos y la unidad de disco si se ha detectado un error. Sólo los usuarios ignorantes o los valientes desesperados ignoran los errores.

Abortar el programa causante del error es una medida drástica, aunque permite salir de una situación que de otra forma podía calificarse de "punto muerto". Por ejemplo, si se asigna erróneamente la impresora a un puerto serie inactivo y se conecta la impresora para que repita (con CONTROL-P), se envía al sistema a una serie infinita de mensajes "impresora fuera de tiempo". Si se aborta el programa, el mecanismo del BIOS que maneja el error ejecuta una función de puesta a cero del sistema (función 0) del BDOS, arranque caliente del CP/M y el control se devuelve al CCP. En el proceso, el conmutador de la impresora se pone a cero y el círculo se rompe.

Manejo práctico de errores

Esta sección trata acerca de algunos errores, describiendo sus causas y la forma en que el BIOS y el usuario pueden manejarlos cuando ocurren.

Errores de entrada/salida de caracteres

A nivel del BIOS, los errores más detectables relacionados con la entrada o salida de caracteres se encontrarán en los chips del hardware.

Error de paridad

Paridad, en este contexto, se refiere al número de bits puestos a 1 en un carácter de 8 bits. El octavo bit no utilizado de otra forma de los caracteres ASCII puede fijarse para hacer este número siempre impar o, alternativa-

mente, siempre par. El hardware de la computadora puede programarse para que cuente el número de bits 1 en cada carácter y para que genere un error si es número impar (paridad impar) o, alternativamente, si es par (paridad par). Si el hardware al otro lado de la línea se programa para operar en la misma forma, el control de paridad proporciona un mecanismo de detección de errores primitivo —puede decirse que un carácter es malo, pero no cuál debería haber sido.

El CP/M no procura un mecanismo estándar para informar de un error de paridad, por lo que la única opción es volver a poner a cero el hardware y colocar un carácter ASCII DEL (7FH, borrado) en el lugar del carácter erróneo.

Si el BIOS opera en un entorno altamente especializado, puede necesitarse contar el número de dichos errores de paridad, de forma que un programa de utilidad pueda informar del rendimiento general del sistema.

Error de construcción

Cuando un carácter ASCII de 8 bits se transmite sobre una línea serie, los ocho bits se transmiten en serie, uno tras otro. Primero se transmite un bit de *arranque,* seguido por el carácter de datos y después por un bit de *stop*. Si el hardware no consigue encontrar los bits de arranque y de stop en las posiciones correctas, ocurrirá un *error de construcción*. Nuevamente, la única opción disponible para el BIOS es poner a cero el chip del hardware y sustituir un ASCII DEL.

Error de exceso de datos

Este error ocurre cuando la introducción de caracteres de datos llega más de prisa de lo que el programa puede manejarlos, de forma que los últimos caracteres se superponen a los que están siendo procesados por el chip de hardware. Este error puede evitarse normalmente con el uso de protocolos de línea serie, como los del BIOS ejemplificado en la figura 8-10.

Un *error de exceso* de datos implica que el protocolo se ha roto. Al igual que con los errores de paridad y de construcción, casi la única opción existente es volver a poner a cero el hardware y sustituir por un carácter DEL.

Error de impresora fuera de tiempo

Este es uno de los pocos errores en los que el BIOS puede intentar sensiblemente una recuperación de error. El error ocurre cuando el BIOS

intenta enviar un carácter a una impresora serie y se encuentra con que la impresora no está dispuesta durante más de 30 segundos, por ejemplo. La causa más corriente de este error es que el usuario olvida poner la impresora en línea. Muchas impresoras requieren estar desconectadas mientras se las alimenta de papel manualmente, y los usuarios olvidan a menudo pulsar el botón para ponerla en línea después.

Después de un retraso de 30 segundos, el BIOS puede enviar un mensaje al dispositivo(s) de la consola informando al usuario del error y pidiendo al usuario que elija el curso de acción adecuado. Obsérvese que la salida de la consola puede dirigirse a más de un dispositivo.

Impresoras paralelo

Las impresoras conectadas al sistema por medio de un puerto paralelo pueden indicar su estado a la computadora mucho más fácilmente que las impresoras serie. Pueden comunicar estados de error tales como "fuera de papel", "final de cinta" y "fuera de línea".

Estos indicadores de un solo error pueden usarse también combinados para indicar si el cable de la impresora está conectado, o bien si la impresora recibe corriente. Se necesita experimentar, poniendo deliberadamente la impresora en estos estados y leyendo su estado para identificarlos. Es equivocado indicar a los usuarios inexpertos que la impresora "está fuera de papel" cuando el problema es que el cable de datos se ha desconectado inadvertidamente.

Sin embargo, cada uno de estos errores puede tratarse de la misma forma que el problema de fuera de tiempo de la impresora serie: componer un mensaje de error y requerir la elección de una forma de acción del usuario.

Ejemplificación de rutina de error de impresora

La figura 9-1 muestra un ejemplo de programa que trata los errores de impresora. Consiste en algunas subrutinas que incluyen:

- La rutina de clasificación e indicación de detección de errores.
- La rutina de corrección de errores.

Utiliza otras subrutinas que se han omitido de la figura para evitar el oscurecimiento de la lógica. Estas subrutinas están listadas totalmente en el BIOS ejemplificado en la figura 8-10.

```
                ;       Este ejemplo muestra en forma explícita cómo manejar el caso
                ;       de que una impresora permanezca ocupada durante demasiado
                ;       tiempo. Con este ejemplo se intenta explicar de forma
                ;       genérica cómo tratar esta clase de errores.
                ;
                ;       El ejemplo presupone la existencia de una interrupción de
                ;       reloj cada 16,666 milisegundos (1/60 segundos) y que el
                ;       control se transferirá a la rutina de reloj de tiempo
                ;       real en cada pulso de reloj.
                ;
                ;       La figura 8-10 muestra un ejemplo más completo instalado
                ;       en un BIOS real.
                ;
0000 =          B$System$Reset          EQU     0       ;Función del BDOS de puesta a cero del sistema
0005 =          BDOS                    EQU     5       ;Punto de entrada al BDOS
                ;
0000 00         Printer$Timeout$Flag:   DB      0       ;Este indicador es puesto a uno por la subrutina
                                                        ;  del servicio de interrupción que es llamada cuando
                                                        ;  el contador de la subrutina de tempor. centinela
                                                        ;  alcanza el 0 (después de haber realizado la
                                                        ;  cuenta atrás durante 30 segundos)
0708 =          Printer$Delay$Count     EQU     1800    ;Con un período de reloj de 16,666 ms
                                                        ;  éste representa un intervalo de 30 segundos
                ;
000D =          CR                      EQU     0DH     ;Retorno de carro
000A =          LF                      EQU     0AH     ;Salto de línea
                ;
                Printer$Busy$Message:
0001 0D0A               DB      CR,LF
0003 5072696E74         DB      'Printer has been busy for too long.',CR,LF
0028 436865636B         DB      'Check that it is on-line and ready.',CR,LF,0
                ;
004E 00         Printer$Character:      DB      0       ;Area de almacenamiento para datos carácter
                                                        ;  a lanzar a salida
                ;
                ;
                LIST:                                   ;<=== Punto de entrada principal del BIOS
                        ;.......                        ;<=== El programa de redirec. de E/S viene aquí
004F 79                 MOV     A,C                     ;Almacenamiento de dato carácter
0050 324E00             STA     Printer$Character
                Printer$Retry:
0053 010807             LXI     B,Printer$Delay$Count   ;Cuenta del número de pulsos de reloj
                                                        ;  antes de llamar a la rutina de
                                                        ;  temporizador centinela
0056 217E00             LXI     H,Printer$Timed$Out     ;  <== dirección
0059 CDA300             CALL    Set$Watchdog            ;Llama a rutina de temporizador centinela
                Printer$Wait:
005C CDA300             CALL    Get$Printer$Status      ;Ver si impresora preparada para aceptar
                                                        ;  salida de caracteres. Incluye la comprobación
                                                        ;  de si está ocupada debido a
                                                        ;  que el controlador está esperando
                                                        ;  un XON, ACK o que DTR se
                                                        ;  haga alto
005F C26C00             JNZ     Printer$Ready           ;Impresora disponible
0062 3A0000             LDA     Printer$Timeout$Flag    ;Comprobación de si contador de centinela
                                                        ;  ha alcanzado el cero (si lo ha hecho,
                                                        ;  la rutina centinela llamará al
                                                        ;  programa Printer$Timed$Out que fija
                                                        ;  este indicador)
0065 B7                 ORA     A
0066 C28400             JNZ     Display$Busy$Message    ;Sí, visualización de mensaje para
                                                        ;  indicar situación de error
0069 C35C00             JMP     Printer$Wait            ;En caso contrario, comprobar si la
                                                        ;  impresora sigue ocupada
                Printer$Ready:                          ;Impresora preparada para escribir carácter,
                                                        ;  pero antes de hacerlo debe ponerse a cero
                                                        ;  el temporizador centinela
006C F3                 DI                              ;Asegurar que no suceda un falso tiempo excedido
006D 010000             LXI     B,0                     ;Se hace poniendo a cero
0070 CDA300             CALL    Set$Watchdog            ;  el contador
```

Figura 9-1. Manejo de error de impresora serie.

```
0073 FB                 EI

0074 3A4E00             LDA     Printer$Character       ;Obtención de carácter para salida
0077 11A300             LXI     D,Printer$Device$Table  ;DE -> tabla de dispositivo para impresora
007A CDA300             CALL    Output$Data$Byte        ;Salida de carácter a la impresora

007D C9                 RET                             ;Retorno al llamador del BIOS
                ;
                ;
                Printer$Timed$Out:                      ;El control llega a este punto desde la
                                                        ;  rutina de temporizador centinela
                                                        ;  si su contador llega a cero.
                                                        ;  Es una rutina del servicio
                                                        ;  de interrupción
                                                        ;Antes de que el control llegue a este punto
                                                        ;  han sido almacenados todos los registros
007E 3EFF               MVI     A,0FFH                  ;Fijación de indicador de tiempo excedido en impresora
0080 320000             STA     Printer$Timeout$Flag
0083 C9                 RET                             ;Retorno a la rutina de temporizador centinela

                                                        ;Rutina de servicio de interrupción
                ;
                Display$Busy$Message:                   ;Impresora ocupada durante más
                                                        ;  de 30 segundos
0084 AF                 XRA     A                       ;Puesta a cero indicador de tiempo excedido
0085 320000             STA     Printer$Timeout$Flag

0088 210100             LXI     H,Printer$Busy$Message  ;Salida de mensaje de error
008B CDA300             CALL    Output$Error$Message

008E CDA300             CALL    Request$User$Choice     ;Visualización de: ¿Nuevo intento?,
                                                        ;  ¿Abortar?, ¿Ignorar? Aceptación
                                                        ;  de un carácter del teclado y retorno
                                                        ;  con el carácter convertido a
                                                        ;  mayúsculas en el registro A
0091 FE52               CPI     'R'                     ;Comprobación de si es Nuevo intento

0093 CA5300             JZ      Printer$Retry
0096 FE41               CPI     'A'                     ;Comprobación de si es Abortar
0098 CA9E00             JZ      Printer$Abort
009B FE49               CPI     'I'                     ;Comprobación de si es Ignorar
009D C8                 RZ
                ;
                Printer$Abort:
009E 0E00               MVI     C,B$System$Reset        ;Lanzamiento de una puesta a cero del sistema
00A0 C30500             JMP     BDOS                    ;No es necesaria la realización de una llamada
                                                        ;  ya que el control no vuelve a este punto
                ;
                ;
                ;       Subrutinas ficticias
                ;       Se muestran para completar en figura 8-10. Los números línea
                ;       en la figura 8-10 se muestran en el campo de comentarios que sigue
                ;
                Printer$Device$Table:           ;Line 01300  (ejemplo de distribución)
                Request$User$Choice:            ;Line 03400
                Output$Error$Message:           ;Line 03500
                Get$Printer$Status:             ;Line 03900  (programa similar)
                Output$Data$Byte:               ;Line 05400  (programa similar)
                Set$Watchdog:                   ;Line 05800
```

Figura 9-1. (Continuación.)

Errores de disco

Los discos son mucho más complicados que los dispositivos de entrada/salida de caracteres. Son posibles los errores en las partes electrónicas y en el propio disco. La mayoría de los errores relacionados con partes electrónicas necesitan solamente ser relacionados con el suficiente detalle

como para proporcionar al ingeniero de mantenimiento información sobre el problema. Esta clase de error es difícilmente corregible reintentando la operación. En contraste, los errores debidos al medio a menudo pueden ser remediados reintentando la operación o mediante la construcción de un software especial de procesamiento de errores en el BIOS. Este capítulo se ocupa de esta clase de errores.

Los errores debidos al medio ocurren cuando el BIOS intenta leer un sector del disco y el hardware detecta un fallo de control de suma en los datos. Esto se conoce como error de *control de redundancia cíclica* (CRC). Algunos controladores de disco realizan un control de lectura después de escritura, de forma que un error CRC puede también ocurrir durante un intento de escribir un sector en el disco.

Con disquetes, el controlador de disco debería reintentar la operación al menos diez veces antes de comunicar el error al usuario. Entonces, como éstos son baratos y reemplazables, el usuario puede elegir el retirar el disquete y continuar con uno nuevo.

Con discos rígidos, el medio no puede cambiarse. La única manera de tratar los sectores malos es reemplazarlos lógicamente, sustituyéndolos por otros sectores que se colocan en su lugar.

Existen dos maneras fundamentales de hacerlo. La figura 9-2 muestra el esquema conocido como "repuesto de sectores" (sector sparing) —sustituyendo sectores de una pista más exterior por sectores que están estropeados.

La ventaja de este esquema es que es dinámico. Si un sector resulta ser malo después de un control lectura-después-de-escritura, incluso después de

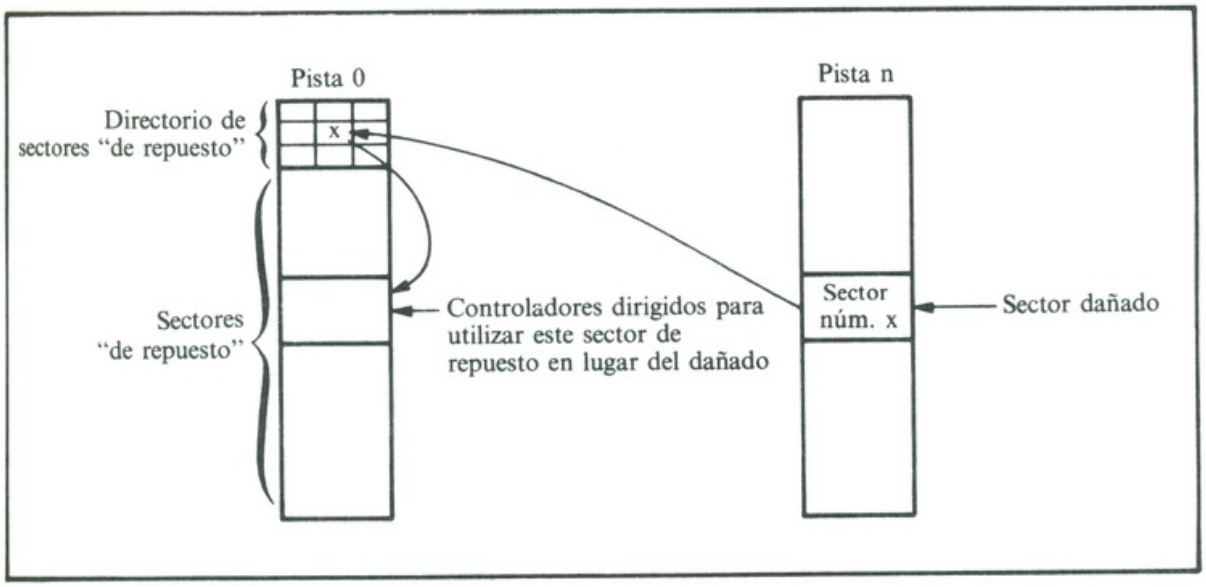


Figura 9-2. Sectores de repuesto.

varios intentos, entonces los datos que se destinaban al sector que falla pueden escribirse en un sector de repuesto. El número del sector que falla se sitúa en un directorio de sectores de repuesto del disco. Después de esto, los controladores de disco volverán a ser dirigidos al sector de repuesto cada vez que se intente leer o escribir en el sector dañado.

El inconveniente de este sistema es que las cabezas de lectura/escritura del disco deberán desplazarse al sector de repuesto y luego volver a la posición anterior para acceder al próximo sector. Esto puede ser un problema si se intenta hacer a gran velocidad en un controlador de cinta de escritura en continuo (streaming tape) (una en la que se escriben datos en forma continua en vez de hacerlo en bloques discretos). El retardo causado por la lectura del sector de repuesto interrumpe el flujo de datos a la unidad de cinta.

Se necesita un programa especial para manipular el directorio de sectores de repuesto, tanto para sustituir un sector que falla de forma manual como para intentar volver a escribir un sector de repuesto de nuevo en el sector dañado.

La figura 9-3 muestra otro esquema para tratar con sectores dañados. En este método, los sectores dañados se saltan en lugar de tener sectores que los sustituyan.

La ventaja de saltar los sectores es que las cabezas no tienen que realizar ninguna indagación larga. El sector que falla se salta, y se utiliza el próximo sector en su lugar. Por ello, el salto de sector puede dar mucho mejor rendimiento. Los datos pueden ser leídos fuera del disco casi con la

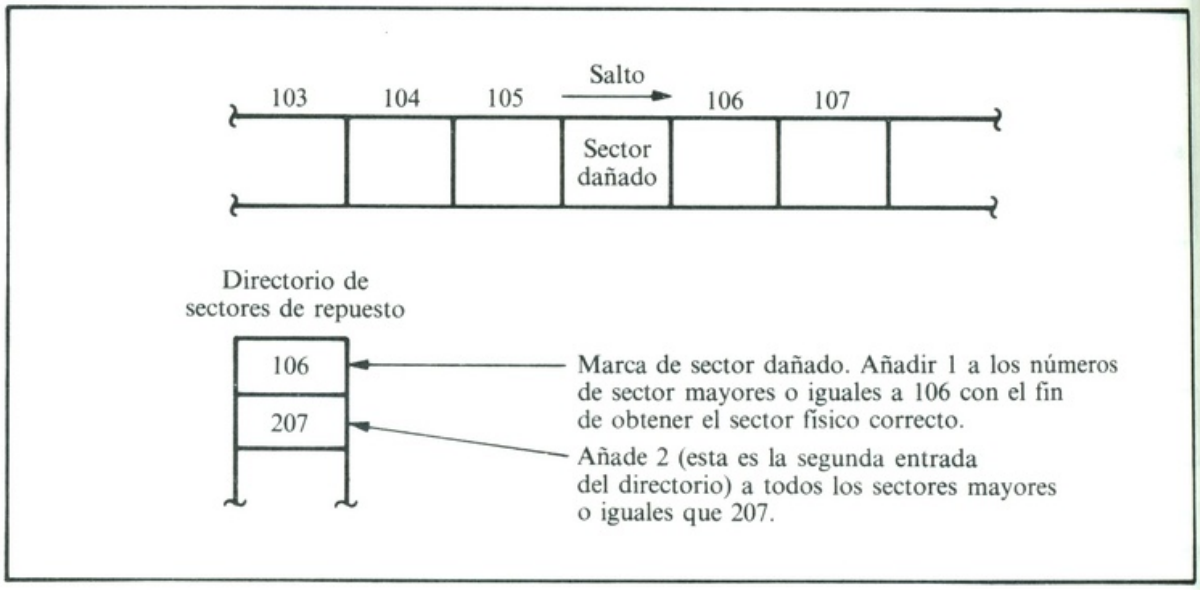


Figura 9-3. Salto de sector.

suficiente velocidad como para mantener la unidad de cinta de escritura en continuo "alimentada" con datos.

El inconveniente de saltar sectores es que no conduce por sí mismo a operaciones dinámicas. La tabla de sectores dañados se construye mejor durante el formateo. Una vez se han escrito los datos en el disco, si un sector se estropea, todos los sectores siguientes del disco deben ser "desplazados uno hacia abajo" con el fin de hacer sitio para el salto del sector dañado. En un disco rígido grande, esto podría llevar varios minutos.

Ejemplo de manejo de sectores dañados

El repuesto y el salto de sectores utilizan una lógica similar. Ambos requieren un directorio de sectores, en cada disco físico, que contenga los números de sector de los sectores dañados. Este directorio se lee en la memoria durante la inicialización del arranque en frío. Después de ello, el disco lee y escribe operaciones referidas a la tabla de memoria residente para ver si están a punto de acceder a un sector dañado.

Para el repuesto de sector, si el sector a punto de ser leído o escrito se encuentra en el directorio de repuesto, su posición en él determina cuál ha de ser el sector que se lea.

En el caso de salto de sector, todos los accesos al disco hacen que el controlador compruebe el directorio de sectores dañados. El directorio se utiliza para saber cuántos sectores dañados existen entre el principio del disco y el sector dañado que ha fallado. Este número deberá ser añadido a la pista y sector requeridos para compensar a todos los sectores dañados.

Los controladores físicos de bajo nivel necesitan cuatro puntos de entrada:

- Leen el sector especificado sin utilizar ningún tipo de procesamiento de sectores dañados. Este se usa para leer en el propio directorio de repuesto.
- Escriben el sector especificado sin utilizar ningún tipo de procesamiento de sectores dañados. Este se usa para escribir en el directorio de repuesto en el disco, tanto para inicializarlo como para actualizarlo.
- Leen y escriben el sector utilizando procesamiento de sectores dañados. Estos puntos de entrada se utilizan para entrada/salida normal del disco.

La figura 9-4 muestra el programa necesario para sectores de repuesto y salto de sector (utilizando una condición inicial).

```
                ;       Este ejemplo muestra las modificaciones a hacer para implementar
                ;       el manejo de sectores erróneos utilizando la técnica de sectores
                ;       de repuesto y el salto de sectores.
                ;
0000 =          False           EQU     0
FFFF =          True            EQU     Not False
                ;
0000 =          Sector$Sparing  EQU     False
FFFF =          Sector$Skipping EQU     Not Sector$Sparing
                ;
                ;
                ;       Equivalencias adicionales y definiciones
                ;
                Spare$Directories:              ;Tabla de direcciones del directorio de repuestos
                                                ;Nota: El propio directorio se
                                                ;  declara al final del
                                                ;  BIOS
0000 D500               DW      Spare$Directory$0       ;Disco físico 0
0002 9701               DW      Spare$Directory$1       ;Disco físico 1
                ;
                Spare$Dir$In$Memory:            ;Indicadores para indicar cuándo se ha cargado en la
0004 00                 DB      0               ;  memoria el directorio de sectores de repuesto de un
0005 00                 DB      0               ;  disco físico dado. Fijado por SELDSK
                ;
                ;
0000 =          Spare$Track             EQU     0       ;Pista conteniendo el directorio de sectores
                                                        ;  de repuesto
0004 =          Spare$Sector            EQU     4       ;Sector que contiene el directorio
0005 =          First$Spare$Sector      EQU     Spare$Sector + 1
                ;
                ;       Variables set by SELDSK
                ;
                Selected$Spare$Directory:
0006 0000                               DW      0       ;Apuntador a directorio
0008 00         Selected$Disk:          DB      0       ;Número de disco lógico
0009 00         Disk$Type:              DB      0       ;Disco flexible/duro
000A 00         Deblocking$Required:    DB      0       ;Indicador de desbloqueo
000B 00         Selected$Physical$Disk: DB      0       ;Número de disco físico
                ;
000C 0000       Disk$Track:     DW      0       ;) Estas variables forman parte del bloque de órdenes
000E 00         Disk$Sector:    DB      0       ;) manejadas por el controlador de disco
                ;
8000 =          Maximum$Track           EQU     32768   ;Utilizado como terminal
0012 =          Sectors$Per$Track       EQU     18
0000 =          First$Sector$On$Track   EQU     0
                ;
                ;
                Disk$Parameter$Headers:
                        ;-------------------------
                        ;Declaraciones DPH estándar
                        ;-------------------------
                ;
                ;
                ;       Equivalencias para bloque de parámetros del disco
                ;
                ;       El byte especial de parámetros del disco que precede a cada
                ;       bloque de parámetros del disco necesita ser redispuesto,
                ;       ya que debe añadirse el número de unidad.
                ;
                ;       Tipos de disco
                ;
                ;                               vvvv--- Número de unidad
0010 =          Floppy$5        EQU     0$001$0000B     ;Minidisquete 5 1/4"
0020 =          Floppy$8        EQU     0$010$0000B     ;Disquete 8" (SC SD)
0030 =          M$Disk          EQU     0$011$0000B     ;Disco "en memoria"
0040 =          H$Disk$10       EQU     0$100$0000B     ;Disco duro - 10 megabyte
                ;
0070 =          Disk$Type$$Mask         EQU     0$111$0000B     ;Máscara para aislar valores
000F =          Physical$Disk$Mask      EQU     0$000$1111B
                ;
                ;       Blocking/deblocking indicator
                ;
0080 =          Need$Deblocking EQU     1$000$0000B     ;Tamaño de sector > 128 bytes
                ;
                ;
                ;       Disk parameter blocks
```

Figura 9-4. Manejo de sectores dañados.

```
                ;
                ;------------------------------
                ;   DPB estándar para A: y B:
                ;------------------------------
                ;
                                                ;Disco lógico C:
                                                ;Byte extra indicando tipo de disco,
                                                ; necesidad de desbloqueo y unidad
                                                ; de disco
000F C0                 DB      H$Disk$10 + Need$Deblocking + 0  ;Unidad física 0
                Hard$5$Parameter$Block$C:
                ;---------------------------------------
                ; Formato estándar del bloque de parámetros
                ;---------------------------------------
                ;
                ;
0010 C0                 DB      H$Disk$10 + Need$Deblocking + 0  ;Unidad física 0
                Hard$5$Parameter$Block$D:
                ;---------------------------------------
                ; Formato estándar del bloque de parámetros
                ;---------------------------------------
                ;
                ;
0004 =          Number$of$Logical$Disks         EQU     4
                ;
                ;
                SELDSK:                 ;Seleccionar disco en registro C
                                        ;C = 0 para disco A, 1 para B, etc.
                                        ;Devuelve la dirección de la cabecera
                                        ; de parámetros del disco en HL o 0000H
                                        ; si no existe el disco seleccionado
                                        ;
0011 210000             LXI     H,0             ;Suposición de error
0014 79                 MOV     A,C             ;Comprobación de disco relecciona válido

0015 FE04               CPI     Number$of$Logical$Disks
0017 D0                 RNC                     ;Retorno si número máximo de disco
0018 320800             STA     Selected$Disk   ;Almacenar número de disco seleccionado
                                                ;Activación para devolver dirección DPH
001B 6F                 MOV     L,A             ;Composición de valor en palabra
001C 2600               MVI     H,0
                                                ;Cálculo del desplazamiento en la tabla
                                                ; cabecera de parámetros del disco multiplicando
                                                ; por la longitud de la cabecera (16 bytes)
001E 29                 DAD     H               ;*2
001F 29                 DAD     H               ;*4
0020 29                 DAD     H               ;*8
0021 29                 DAD     H               ;*16
0022 110F00             LXI     D,Disk$Parameter$Headers        ;Obtención dirección base
0025 19                 DAD     D               ;DE -> DPH apropiado
0026 E5                 PUSH    H               ;Salvar dirección de DPH
                                                ;
                                                ;Acceso al bloque de parámetros del disco
                                                ; para extraer el byte especial de prefijo
                                                ; que identifica el tipo de disco
                                                ; y si se precisa desbloqueo
                                                ;
0027 110A00             LXI     D,10            ;Obtención desplazamiento apuntador a DPB en DPH
002A 19                 DAD     D               ;DE -> dirección DPB en DPH
002B 5E                 MOV     E,M             ;Obtención dirección DPB en DE
002C 23                 INX     H
002D 56                 MOV     D,M
002E EB                 XCHG                    ;DE -> DPB

                SELDSK$Set$Disk$Type:
002F 2B                 DCX     H               ;DE -> byte prefijo
0030 7E                 MOV     A,M             ;Obtención byte prefijo
0031 E670               ANI     Disk$Type$Mask  ;Aislar tipo de disco
0033 320900             STA     Disk$Type       ;Almac. para utilización en controlador de bajo nivel
0036 7E                 MOV     A,M             ;Obtención de otra copia del byte prefijo
0037 E680               ANI     Need$Deblocking         ;Aislar indicador de desbloqueo
0039 320A00             STA     Deblocking$Required     ;Salvar para utilización en controlador de bajo
                                                        ;  nivel
                                                ;Programa adicional para comprobar si ha sido
                                                ;  leído el directorio de rectores de repuesto
                                                ;  para un disco dado
```

Figura 9-4. (Continuación.)

```
003C 7E                 MOV     A,M             ;Obtención de número de disco físico
003D E60F               ANI     Physical$Disk$Mask
003F 320B00             STA     Selected$Physical$Disk  ;Salvar para controladores de bajo nivel

0042 5F                 MOV     E,A             ;Puesta en palabra
0043 1600               MVI     D,0
0045 210400             LXI     H,Spare$Dir$In$Memory   ;Puesta de apuntador en tabla
0048 19                 DAD     D

0049 7E                 MOV     A,M             ;Obtención de indicador
004A B7                 ORA     A
004B C27700             JNZ     Dir$In$Memory   ;Directorio "de repuesto" ya en memoria
004E 34                 INR     M               ;Fijar indicador

004F 210000             LXI     H,Spare$Directories     ;Creación de apuntador
0052 19                 DAD     D                       ;  a directorio "de repuesto"
0053 19                 DAD     D                       ;  (añadido dos veces)

                                                        ;HL -> palabra conteniendo dirección del directorio
0054 5E                 MOV     E,M
0055 23                 INX     H
0056 56                 MOV     D,M                     ;Dirección del directorio de repuestos en DE
0057 EB                 XCHG                            ;HL -> directorio de repuestos

0058 220600             SHLD    Selected$Spare$Directory ;Guardar para utilización por los controladores
                                                        ;  físicos más adelante

005B 110000             LXI     D,Spare$Track           ;Pista conteniendo el directorio de repuestos
005E 3A0B00             LDA     Selected$Physical$Disk
0061 47                 MOV     B,A
0062 3E04               MVI     A,Spare$Sector          ;Sector que contiene el directorio de repuestos
0064 0E18               MVI     C,Spare$Length/8        ;Número de bytes del directorio de repuestos / 8
0066 CDD500             CALL    Absolute$Read           ;Lectura en el directorio de repuestos - sin
                                                        ;  utilizar la administración de sectores erróneos
0069 2A0600             LHLD    Selected$Spare$Directory ;Fijar marca de final
006C 11C000             LXI     D,Spare$Length          ;  al final del directorio de repuestos
006F 19                 DAD     D
0070 110080             LXI     D,Maximum$Track         ;Utilización del número máximo de pista
0073 73                 MOV     M,E
0074 23                 INX     H
0075 3602               MVI     M,D

                Dir$In$Memory:
0077 E1                 POP     H               ;Recuperación apuntador a DPH
0078 C9                 RET
                ;
                ;
                ;       En los controladores de disco de bajo nivel puede insertarse
                ;       el siguiente programa antes de activar el controlador para ejecutar
                ;       una orden de lectura o de escritura.
                ;
0079 2A0C00             LHLD    Disk$Track              ;Obtención del número de pista de la tabla
                                                        ;  de órdenes del controlador de disco
007C EB                 XCHG                            ;DE = pista
007D 2A0600             LHLD    Selected$Spare$Directory ;HL -> directorio de repuestos
0080 2B                 DCX     H                       ;Vuelta a entrada anterior
0081 2B                 DCX     H                       ;  (3 bytes)
0082 2B                 DCX     H

0083 3A0E00             LDA     Disk$Sector             ;Obtención número de sector
0086 4F                 MOV     C,A                     ;Salvar para uso posterior

0087 06FF               MVI     B,0FFH                  ;Fijar contador

                Check$Next$Entry:
0089 23                 INX     H                       ;Actualización para entrada siguiente (o primera)
                Check$Next$Entry1:
008A 23                 INX     H
                Check$Next$Entry2:
008B 23                 INX     H

008C 04                 INR     B                       ;Actualización contador

                        IF      Sector$Sparing
                                                        ;Si se utiliza la técnica de sectores de
                                                        ;  repuesto, se indica el final de la tabla
```

Figura 9-4. (Continuación.)

```
                                                    ;  con una entrada con el número de pista
                                                    ;  = al número máximo de pista
                        LXI     D,Maximum$Track     ;Obtención número máximo de pista
                        CALL    CMPM                ;Comparación DE con (HL), (HL+1)
                        JZ      Not$Bad$Sector      ;Final de tabla alcanzado
                        ENDIF

                                                    ;Nota: Si se utiliza la técnica de salto
                                                    ;  de sectores, el bucle de búsqueda siguiente
                                                    ;  terminará cuando la pista requerida
                                                    ;  es menor que la de la tabla
                                                    ;Ello ocurrirá siempre cuando se encuentra
                                                    ;  el número máximo de pista al final
                                                    ;  de la tabla

008D EB                 XCHG                        ;DE -> entrada de tabla
008E 2A0C00             LHLD    Disk$Track          ;Obtención pista requerida
0091 EB                 XCHG                        ;DE = pista requerida, HL -> entrada de tabla
0092 CDCD00             CALL    CMPM                ;Comparación de la pista requerida con la entrada
                                                    ;  de tabla

                        IF      Sector$Sparing      ;Utilícese el programa siguiente
                                                    ;  para sectores de repuesto
                        JNZ     Check$Next$Entry    ;Pista no coincidente
                        INX     H                   ;HL -> byte MS de la pista
                        INX     H                   ;HL -> sector
                        MOV     A,C                 ;Obtención sector requerido
                        CMP     M                   ;Comparación con entrada de tabla
                        JNZ     Check$Next$Entry2   ;Sector no coincidente

                                                    ;Pista y sector coincidentes,
                                                    ;  sustituir, por tanto, pista de repuesto
                                                    ;  y sector apropiado

                        LXI     H,Spare$Track       ;Obtención número de pista utilizado para
                                                    ;  sectores de repuesto
                        SHLD    Disk$Track          ;Sustitución de pista

                        MVI     A,First$Spare$Sector ;Obtención primer número de sector
                        ADD     B                   ;Añadir el número de entrada al
                                                    ;  directorio que corresponda
                        STA     Disk$Sector         ;Sustitución de sector
                        ENDIF

                        IF      Sector$Skipping     ;Utilícese el siguiente programa para
                                                    ;  salto de sectores
                                                    ;Su objeto es encontrar la entrada
                                                    ;  en la tabla que es mayor
                                                    ;  o igual que el sector/pista
                                                    ;  requeridos

0095 CA9E00             JZ      Tracks$Match        ;Posible coincidencia para pista y sector
0098 D2AC00             JNC     Compute$Increment   ;Pista requerida < entrada de tabla
009B C38900             JMP     Check$Next$Entry    ;Pista requerida > entrada de tabla

                Tracks$Match:
009E 23                 INX     H                   ;HL -> byte MS de la pista
009F 23                 INX     H                   ;HL -> sector
00A0 77                 MOV     M,A                 ;obtención de sector a partir de la tabla

00A1 B9                 CMP     C                   ;Comparación con sector requerido
00A2 CAAB00             JZ      Sectors$Match       ;Coincidencia pista/sector
00A5 D2AC00             JNC     Compute$Increment   ;Pista/sector requeridos < pista/sector de repuesto
00A8 C3B800             JMP     Check$Next$Entry2   ;Trasladar a entrada de tabla siguiente

                Sectors$Match:                      ;Si coinciden pista y sector con entrada
00AB 04                 INR     B                   ;  de tabla, puede saltarse entonces
                                                    ;  el sector adicional

                Compute$Increment:                  ;B contiene el número acumulativo de
                                                    ;  sectores a saltar
                                                    ;Obtención sector requerido
00AC 79                 MOV     A,C                 ;Saltar número de sectores requerido
00AD 80                 ADD     B                   ;Determinar número de sector final
00AE 0612               MVI     B,Sectors$Per$Track ;  e incremento de pista
```

Figura 9-4. (Continuación.)

```
00B0 CDC300             CALL    DIV$A$BY$B              ;Devolver C = cociente, A = resto
00B3 320E00             STA     Disk$Sector             ;A = nuevo número de sector

00B6 59                 MOV     E,C                     ;Componer palabra con incremento de pista
00B7 1600               MVI     D,0
00B9 2A0C00             LHLD    Disk$Track              ;Obtención de pista requerida
00BC 19                 DAD     D                       ;Añadir con incremento
00BD 220C00             SHLD    Disk$Track              ;Almacenar pista actualizada
                        ENDIF

                Not$Bad$Sector:
                                                        ;Pista/sector no erróneos
                                                        ;  o pista y sector han sido
                                                        ;  actualizados
00C0 C3D500             JMP     Read$Write$Disk         ;Realizar lectura/escritura de disco físico
                ;
                        IF      Sector$Skipping
                                                        ;Subrutina necesaria para la rutina
                                                        ;  de salto de sectores
                ;
                ;
                ;       DIV$A$BY$B
                ;       Dividir A entre B
                ;
                ;       Esta rutina divide A entre B, devolviendo el cociente
                ;       en C y el resto en A.
                ;
                ;       Parámetros de entrada
                ;
                ;               A = dividendo
                ;               B = divisor
                ;
                ;       Parámetros de salida
                ;
                ;               A = resto
                ;               B = cociente
                ;
                DIV$A$BY$B:
00C3 0E00               MVI     C,0             ;Inicializar cociente
                DIV$A$BY$B$Loop:
00C5 0C                 INR     C               ;Incrementar cociente
00C6 90                 SUB     B               ;Sustraer divisor

00C7 F2C500             JP      DIV$A$BY$B$Loop ;Repetir si resultado todavía positivo
00CA 0D                 DCR     C               ;Cociente correcto
00CB 80                 ADD     B               ;Resto correcto
00CC C9                 RET
                        ENDIF
                ;
                ;       CMPM
                ;       Comparación memoria
                ;
                ;       Esta subrutina compara el contenido de DE con (HL) y (HL+1),
                ;       devolviendo los indicadores como si la sustracción (HL) - DE
                ;       se hubiese realizado.
                ;
                ;       Parámetros de entrada
                ;
                ;               HL -> palabra en memoria
                ;               DE = valor a comprobar
                ;
                ;       Parámetros de salida
                ;
                ;               Indicadores fijados por (HL) - DE
                ;
                CMPM:
00CD 7E                 MOV     A,M                     ;Obtención byte MS
00CE BA                 CMP     D
00CF C0                 RNZ                             ;Retorno si bytes MS distintos
00D0 23                 INX     H                       ;HL -> byte LS
00D1 7E                 MOV     A,M                     ;Obtención byte LS
00D2 BB                 CMP     E
00D3 2B                 DCX     H                       ;Retorno sin cambiar HL
00D4 C9                 RET

                ;
```

Figura 9-4. (Continuación.)

```
                ;
                Absolute$Read:
                                        ;Las rutinas de lectura (y escritura) absoluta acceden
                                        ;  al sector y pista especificada sin utilizar ningún tipo
                                        ;  de administración de sectores erróneos.
                ;       Parámetros de entrada
                ;
                ;               HL -> memoria intermedia
                ;               DE =  pista
                ;               A = sector
                ;               B = número de unidad de disco
                ;               C = número de bytes a leer / 8
                ;
                ;       Fijar bloque de comandos del controlador de disco con
                ;       parámetros situados en registros, inicializar entonces la
                ;       operación de lectura yendo al programa Read$Write$Disk.
                ;
                ;
                Read$Write$Disk:

                        ;-------------------------------------------------
                        ; El resto del programa de los controladores de bajo nivel
                        ;   se sigue, leyendo la pista y sector requeridos.
                        ;-------------------------------------------------

                ;
                ;       Declaraciones del directorio de repuestos
                ;
                ;       Nota: La utilidad de formateado de disco crea un directorio
                ;       inicial de repuestos con entradas de pista/sector
                ;       para aquellos sectores que encuentra erróneos. Llena el
                ;       resto del directorio con 0FFH (lo que sirve para terminar
                ;       la búsqueda en el directorio).
                ;
                ;
00C0 =          Spare$Length    EQU     64 * 3          ;64 entradas, 3 bytes cada una
                                                        ;  Byte 0,1 = pista
                                                        ;  Byte 2 = sector

                Spare$Directory$0:
00D5                    DS      Spare$Length    ;Directorio de repuestos
0195                    DS      2               ;Fijación por SELDSK del número máximo de pista
                                                ;  para mayor seguridad. La utilidad FORMAT pone
                                                ;  el número máximo de pista en las entradas
                                                ;  no utilizadas del directorio de repuestos.

                Spare$Directory$1:
0197                    DS      Spare$Length    ;Directorio de repuestos propiamente dicho
0257                    DS      2               ;Marca de final
```

Figura 9-4. (Continuación.)

Mejora de mensajes de error

La ampliación final al manejo de errores del BIOS que se trata aquí se encuentra en el manejo de mensajes de error del controlador de disco. La subrutina mostrada en el BIOS ejemplo de la figura 8-10, aunque es una mejora significativa de los mensajes enviados normalmente por el BDOS, no advierte al usuario del curso de acción más adecuado para cada error. La figura 9-5 muestra una versión mejorada del procesador de mensajes de error.

```
                ;       Muestra muy brevemente el procesador de errores con
                ;       ayuda-al-usuario para los errores de disco que se han
                ;       mostrado en el BIOS mejorado de la figura 8-10.
                ;       Esta versión presenta unas recomendaciones de la acción
                ;       a seguir dependiendo de la naturaleza del error detectado.
                ;       Las partes que permanecen del programa de la figura 8-10
                ;       se han abreviado.
                ;
                ;       Equivalencias ficticias y declaraciones de datos necesarias
                ;       para obtener un ensamblaje de errores de este ejemplo.
                ;
0001 =          Floppy$Read$Code        EQU     01H     ;Lectura de orden para controlador
0002 =          Floppy$Write$Code       EQU     02H     ;Escritura de orden para controlador
                ;
0000 00         Disk$Hung$Flag:         DB      0       ;Fijar NZ cuando temporizador centinela
                                                        ;  fuera del límite
0258 =          Disk$Timer              EQU     600     ;Retardo de 10 segundos (16,66 ms pulsos)
                ;
0043 =          Disk$Status$Block       EQU     43H     ;Dirección de memoria donde devolver estado
                                                        ;  del controlador
                                                        ;Valores de la tabla de órdenes del controlador
0001 00         Floppy$Command:         DB      0
0002 00         Floppy$Head:            DB      0
0003 00         Floppy$Track:           DB      0
0004 00         Floppy$Sector:          DB      0
0005 00         Deblocking$Required:    DB      0       ;Fijación de indicador por SELDSK de acuerdo
                                                        ;  con el tipo de disco seleccionado

0006 00         Disk$Error$Flag:        DB      0       ;Indicador de error devuelto al BDOS
                ;
0007 00         In$Buffer$Disk:         DB      0       ;Idem de disco en relación con el sector de disco
                                                        ;  en curso en la memoria intermedia de desbloqueo
                ;
                ;       Equivalencias para mensajes
                ;
0007 =          BELL    EQU     07H     ;Indicador sonoro del terminal
000D =          CR      EQU     0DH     ;Retorno de carro
000A =          LF      EQU     0AH     ;Salto de línea
                ;
0005 =          BDOS    EQU     5       ;Punto de entrada al BDOS (para puesta a cero del sistema)
                ;
                ;
                ;
                ;
                No$Deblock$Retry:
                        ;-------------------------------------------------
                        ; Programa omitido para fijar la tabla de órdenes del
                        ; controlador de disco e iniciar la operación de disco
                        ;-------------------------------------------------
0008 C31500             JMP     Wait$For$Disk$Complete
                ;
                ;
                Write$Physical:                 ;Escritura del contenido de la memoria intermedia de disco
                                                ;  en el sector correcto
000B 3E02               MVI     A,Floppy$Write$Code     ;Obtención código de función de escritura

000D C31200             JMP     Common$Physical ;Ir a programa común
                Read$Physical:                  ;Lectura del sector seleccionado previamente
                                                ;  en la memoria intermedia de disco
0010 3E01               MVI     A,Floppy$Read$Code      ;Obtención código de función de lectura
                Common$Physical:
0012 320100             STA     Floppy$Command  ;Fijar tabla de órdenes

                ;
                Deblock$Retry:                  ;Punto de reentrada para intento después de error
                        ;-------------------------------------------------
                        ; Programa omitido para fijar el bloque de órdenes del
                        ; controlador de disco e iniciar la operación de disco
                        ;-------------------------------------------------
                ;
                Wait$For$Disk$Complete:         ;Espera hasta que el bloque de estado del disco
                                                ;  indique que se ha completado la operación,
                                                ;  entonces comprobación de ocurrencia de errores
                                                ;En entrada, HL -> byte de control de disco
0015 AF                 XRA     A                       ;Asegurar borrado del indicador de suspensión
```

Figura 9-5. Procesador de errores de disco de ayuda al usuario.

```
0016 320000             STA     Disk$Hung$Flag

0019 213100             LXI     H,Disk$Timed$Out        ;Activar temporizador centinela
001C 015802             LXI     B,Disk$Timer            ;Retardo de tiempo
001F CD3B03             CALL    Set$Watchdog
                Disk$Wait$Loop:
0022 7E                 MOV     A,M                     ;Obtención byte de control
0023 B7                 ORA     A
0024 CA3700             JZ      Disk$Complete           ;Operación realizada

0027 3A0000             LDA     Disk$Hung$Flag          ;Comprobación de tiempo excedido
002A B7                 ORA     A
002B C29F02             JNZ     Disk$Error              ;Se fijará a 40H

002E C32200             JMP     Disk$Wait$Loop

                Disk$Timed$Out:                 ;La rutina de temporizador centinela devuelve el
                                                ;  control a este punto -- por tanto, esto forma parte
                                                ;  de la rutina de servicio de interrupción
0031 3E40               MVI     A,40H                   ;Fijar código de error de suspensión
0033 320000             STA     Disk$Hung$Flag          ;  de disco en el indicador de error
                                                        ;  para control fuera del bucle
0036 C9                 RET                             ;Retorno a rutina de temporizador centinela

                Disk$Complete:
0037 010000             LXI     B,0                     ;Puesta a cero temporizador centinela
                                                        ;HL irrelevante
003A CD3B03             CALL    Set$Watchdog

003D 3A4300             LDA     Disk$Status$Block       ;Completar -- comprobación de estado
0040 FE80               CPI     80H                     ;Comprobación de ocurrencia de errores
0042 DA9F02             JC      Disk$Error              ;Sí
                ;
                Disk$Error$Ignore:
0045 AF                 XRA     A                       ;No
0046 320600             STA     Disk$Error$Flag         ;Borrado de indicador de error
0049 C9                 RET

                ;
                ;       Manejo de mensajes de error del disco
                ;
                ;
                Disk$Error$Messages:            ;Se examina esta tabla comprobando el estado
                                                ;  de error del disco con los de la tabla. Cuando
                                                ;  hay coincidencias o se ha alcanzado
                                                ;  el final de la tabla, la dirección
                                                ;  que sigue al valor del estado apunta
                                                ;  al texto del mensaje de aviso correcto.
                                                ;  Inmediatamente después está la dirección de un
                                                ;  mensaje de descripción del error.
004A 40                 DB      40H
004B B0019500           DW      Disk$Advice1,Disk$Msg$40
004F 41                 DB      41H
0050 C9019A00           DW      Disk$Advice2,Disk$Msg$41
0054 42                 DB      42H
0055 E301A400           DW      Disk$Advice3,Disk$Msg$42
0059 21                 DB      21H
005A 0702B400           DW      Disk$Advice4,Disk$Msg$21
005E 22                 DB      22H
005F 1B02B900           DW      Disk$Advice5,Disk$Msg$22
0063 23                 DB      23H
0064 1B02C000           DW      Disk$Advice5,Disk$Msg$23
0068 24                 DB      24H
0069 3D02D200           DW      Disk$Advice6,Disk$Msg$24
006D 25                 DB      25H
006E 3D02DE00           DW      Disk$Advice6,Disk$Msg$25
0072 11                 DB      11H
0073 5302F100           DW      Disk$Advice7,Disk$Msg$11
0077 12                 DB      12H
0078 5302FF00           DW      Disk$Advice7,Disk$Msg$12
007C 13                 DB      13H
007D 53020C01           DW      Disk$Advice7,Disk$Msg$13
0081 14                 DB      14H
0082 53021A01           DW      Disk$Advice7,Disk$Msg$14
0086 15                 DB      15H
0087 53022901           DW      Disk$Advice7,Disk$Msg$15
```

Figura 9-5. (Continuación.)

```
008B 16                 DB      16H
008C 53023501           DW      Disk$Advice7,Disk$Msg$16
0090 00                 DB      0                       ;<== Terminal
0091 53024501           DW      Disk$Advice7,Disk$Msg$Unknown   ;Código sin coincidencia
                ;
0005 =          DEM$Entry$Size  EQU     5       ;Entrar tamaño en tabla de mensaje de error
                ;
                ;
                ;       Message texts
                ;
0095 48756E6700Disk$Msg$40:     DB      'Hung',0        ;Mensaje de tiempo excedido
009A 4E6F742052Disk$Msg$41:     DB      'Not Ready',0
00A4 5772697465Disk$Msg$42:     DB      'Write Protected',0
00B4 4461746100Disk$Msg$21:     DB      'Data',0
00B9 466F726D61Disk$Msg$22:     DB      'Format',0
00C0 4D69737369Disk$Msg$23:     DB      'Missing Data Mark',0
00D2 4275732054Disk$Msg$24:     DB      'Bus Timeout',0
00DE 436F6E7472Disk$Msg$25:     DB      'Controller Timeout',0
00F1 4472697665Disk$Msg$11:     DB      'Drive Address',0
00FF 4865616420Disk$Msg$12:     DB      'Head Address',0
010C 547261636BDisk$Msg$13:     DB      'Track Address',0
011A 536563746FDisk$Msg$14:     DB      'Sector Address',0
0129 4275732041Disk$Msg$15:     DB      'Bus Address',0
0135 496C6C6567Disk$Msg$16:     DB      'Illegal Command',0
0145 556E6B6E6FDisk$Msg$Unknown:        DB      'Unknown',0
                ;
                Disk$EM$1:
                                                ;Mensaje de error disco principal -- parte 1
014D 070D0A                     DB      BELL,CR,LF
0150 4469736B20                 DB      'Disk ',0
                ;
                                                ;Siguiente salida texto de error
                ;
                Disk$EM$2:
                                                ;Mensaje de error disco principal -- parte 2
0156 204572726F                 DB      ' Error ('
015E 0000       Disk$EM$Status: DB      0,0     ;Código de estado hexadecimal
0160 290D0A2020                 DB      ')',CR,LF,'     Drive '
016E 00         Disk$EM$Drive:  DB      0       ;Código de unidad de disco A,B...
016F 2C20486561                 DB      ', Head '
0176 00         Disk$EM$Head:   DB      0       ;Número de cabeza
0177 2C20547261                 DB      ', Track '
017F 0000       Disk$EM$Track:  DB      0,0     ;Número de pista
0181 2C20536563                 DB      ', Sector '
018A 0000       Disk$EM$Sector: DB      0,0     ;Número de sector
018C 2C204F7065                 DB      ', Operation - '
019A 00                         DB      0               ;Terminal
                ;
019B 526561642EDisk$EM$Read:    DB      'Read.',0       ;Nombres de operación
01A1 5772697465Disk$EM$Write:   DB      'Write.',0
                ;
01A8 0D0A202020Disk$Advice0:    DB      CR,LF,'     ',0
01B0 436865636BDisk$Advice1:    DB      'Check disk loaded, Retry',0
01C9 506F737369Disk$Advice2:    DB      'Possible hardware problem',0
01E3 5772697465Disk$Advice3:    DB      'Write enable if correct disk, Retry',0
0207 5265747279Disk$Advice4:    DB      'Retry several times',0
021B 5265666F72Disk$Advice5:    DB      'Reformat disk or use another disk',0
023D 4861726477Disk$Advice6:    DB      'Hardware error, Retry',0
0253 4861726477Disk$Advice7:    DB      'Hardware or Software error, Retry',0
                ;
0275 2C206F7220Disk$Advice9:    DB      ', or call for help if error persists',CR,LF
                ;
                Disk$Action$Confirm:
029B 00                         DB      0       ;Fijado a carácter entrado por el usuario
029C 0D0A00                     DB      CR,LF,0
                ;
                ;       Procesador de errores de disco
                ;
                ;       Esta rutina construye y lanza un mensaje de error.
                ;       Se plantea entonces al usuario la oportunidad de elegir entre:
                ;
                ;               R -- Intentar de nuevo la operación que ha causado el error
                ;               I -- Ignorar el error e intentar la continuación
                ;               A -- Abortar el programa y retornar a CP/M
                ;
                Disk$Error:
029F F5                 PUSH    PSW             ;Preservar código de error desde controlador
```

Figura 9-5. (Continuación.)

```
02A0 215E01             LXI     H,Disk$EM$Status        ;Conversión del código del mensaje
02A3 CD3B03             CALL    CAH                     ;Conversión de A a hexadecimal

02A6 3A0700             LDA     In$Buffer$Disk          ;Conversión de identificador de disco para mensaje
02A9 C641               ADI     'A'                     ;Composición de la letra correspondiente
02AB 326E01             STA     Disk$EM$Drive

02AE 3A0200             LDA     Floppy$Head             ;Conversión del número de cabeza
02B1 C630               ADI     '0'
02B3 327601             STA     Disk$EM$Head

02B6 3A0300             LDA     Floppy$Track            ;Conversión del número de pista
02B9 217F01             LXI     H,Disk$EM$Track
02BC CD3B03             CALL    CAH

02BF 3A0400             LDA     Floppy$Sector           ;Conversión del número de sector
02C2 218A01             LXI     H,Disk$EM$Sector
02C5 CD3B03             CALL    CAH

02C8 214D01             LXI     H,Disk$EM$1             ;Salida de la primera parte del mensaje
02CB CD3B03             CALL    Output$Error$Message

02CE F1                 POP     PSW                     ;Recuperación código de mensaje de error
02CF 47                 MOV     B,A                     ;Para comparaciones
02D0 214500             LXI     H,Disk$Error$Messages - DEM$Entry$Size
                                                        ;HL -> tabla -- en entrada
02D3 110500             LXI     D,DEM$Entry$Size        ;Obtención de tamaño de la entrada para bucle
                                                        ;  posterior
                Disk$Error$Next$Code:
02D6 19                 DAD     D                       ;Movimiento a la siguiente (o primera) entrada

02D7 7E                 MOV     A,M                     ;Obtención número de código de la tabla
02D8 B7                 ORA     A                       ;Comprobación de final de tabla
02D9 CAE302             JZ      Disk$Error$Matched      ;Sí, coincidencia
02DC B8                 CMP     B                       ;Comparar a código real
02DD CAE302             JZ      Disk$Error$Matched      ;Sí, salida del bucle
02E0 C3D602             JMP     Disk$Error$Next$Code    ;Comprobación código siguiente
                ;
                Disk$Error$Matched:
02E3 23                 INX     H                       ;HL -> dirección mensaje de aviso
02E4 5E                 MOV     E,M
02E5 23                 INX     H
02E6 56                 MOV     D,M                     ;DE -> test de aviso
02E7 D5                 PUSH    D                       ;Almacenar para utilización posterior

02E8 23                 INX     H                       ;HL -> dirección de texto mensaje
02E9 5E                 MOV     E,M                     ;Obtención de la dirección en DE
02EA 23                 INX     H
02EB 56                 MOV     D,M

02EC EB                 XCHG                            ;HL -> texto
02ED CD3B03             CALL    Output$Error$Message    ;Visualización texto explicativo

02F0 215601             LXI     H,Disk$EM$2             ;Visualización segunda parte del mensaje
02F3 CD3B03             CALL    Output$Error$Message

02F6 219B01             LXI     H,Disk$EM$Read          ;Elección operación de texto
                                                        ;  (suposición de lectura)
02F9 3A0100             LDA     Floppy$Command          ;Obtención orden de controlador
02FC FE01               CPI,    Floppy$Read$Code
02FE CA0403             JZ      Disk$Error$Read         ;Sí
0301 21A101             LXI     H,Disk$EM$Write         ;No, cambio de dirección en HL
                Disk$Error$Read:
0304 CD3B03             CALL    Output$Error$Message    ;Visualización tipo de operación

0307 21AB01             LXI     H,Disk$Advice0          ;Visualización blancos encabezamiento
030A CD3B03             CALL    Output$Error$Message

030D E1                 POP     H                       ;Recuperación apuntador a texto
030E CD3B03             CALL    Output$Error$Message

0311 217502             LXI     H,Disk$Advice9          ;Visualización segunda componente
0314 CD3B03             CALL    Output$Error$Message
                ;
```

Figura 9-5. (Continuación.)

```
                Disk$Error$Request$Action:              ;Pregunta usuario acción siguiente
0317 CD3B03             CALL    Request$User$Choice     ;Visualización de indicación y espera
                                                        ;  para entrar retorno con A = carácter
                                                        ;  en mayúscula
031A FE52               CPI     'R'                     ;¿Nuevo intento?
031C CA2C03             JZ      Disk$Error$Retry
031F FE41               CPI     'A'                     ;Abortar
0321 CA3603             JZ      System$Reset
0324 FE49               CPI     'I'                     ;Ignorar
0326 CA4500             JZ      Disk$Error$Ignore
0329 C31703             JMP     Disk$Error$Request$Action
                ;
                Disk$Error$Retry:                       ;La decisión de dónde volver depende de
                                                        ;  si la operación fallida ha
                                                        ;  ocurrido en una unidad desbloqueada
                                                        ;  o no desbloqueada
032C 3A0500             LDA     Deblocking$Required
032F B7                 ORA     A
0330 C21500             JNZ     Deblock$Retry
0333 C30800             JMP     No$Deblock$Retry
                ;
                System$Reset:                           ;Esto es una decisión radical, pero
                                                        ;  causa el reinicio de CP/M
0336 0E00               MVI     C,0                     ;Reinicialización del sistema
0338 CD0500             CALL    BDOS

                ;
                ;       Las subrutinas omitidas están completas en la figura 8-10
                ;
                Set$Watchdog:           ;Fijar temporizador centinela (al número de pulsos en BC y para
                                        ;  transferir control a (HL) cuando el temporizador alcanza el cero)
                CAH:                    ;Conversión de A a dos caracteres hexadecimales ASCII, almacenando
                                        ;  la salida en (HL) y (HL+1)
                Output$Error$Message:   ;Visualización del mensaje de error terminado por byte 00H
                                        ;  apuntado por HL. El mensaje se dirige
                                        ;  a las consolas que no se están usando
                                        ;  además como impresoras.
                Request$User$Choice:    ;Visualizar inidicación "Entrada R, A, I..." y devolución
                                        ;  de un único carácter en la consola (en mayúscula) en A
033B C9                 RET             ;Ficticio
```

Figura 9-5. (Continuación.)

Técnicas básicas de depuración
Subrutinas de depuración
Herramientas software para depuración
Puesta en marcha del CP/M por primera vez
Depuración del cargador del CP/M
Depuración del BIOS
Comprobación en vivo de un nuevo BIOS

10

Comprobación detenida de un nuevo BIOS

Este capítulo se enfrenta con algunos de los problemas que se presentarán al montar el CP/M en una computadora por primera vez o al mejorarlo una vez montado y funcionando en ella.

En el primer caso, cuando el CP/M aún no funciona en la computadora, puede escribirse uno mismo el BIOS completo, utilizando como modelo el BIOS ejemplo que se suministra en el disquete del CP/M y el código de ejemplo del capítulo 6.

En el segundo caso se puede ampliar el BIOS existente añadiendo nuevo código —de los ejemplos de los capítulos 8 y 9, programas obtenidos en revistas de informática o creados por uno mismo—. Para hacer esto se necesitará tener acceso al código fuente del BIOS —existe un problema, si los fabricantes de la computadora no lo dejan disponible—. En general, sin embargo, el código fuente del BIOS está incluido en el sistema o puede obtenerse a un coste nominal o sin él. Si no se puede obtener el código fuente, se puede, ciertamente, tomar el toro por los cuernos y reimplemen-

tar el CP/M en el propio sistema. Esto puede requerir muchas horas de desensamblamiento del código máquina del BIOS actual para encontrar la forma de acceder a todos los puertos diferentes y cómo controlar los dispositivos a los que están conectados.

Aunque el BIOS es el componente más importante de una nueva implementación del CP/M, recuérdese que sólo es el comienzo —se puede utilizar la misma cantidad de tiempo y esfuerzo obteniendo el cargador de todas las utilidades.

Técnicas básicas de depuración

Antes de sumergirnos en los detalles de cómo se depura una implementación del CP/M es útil considerar la naturaleza de la tarea. Se imponen aquí algunas observaciones:

> "La comprobación de programas puede utilizarse para mostrar la presencia de errores, pero nunca para mostrar su ausencia."—Dijkstra
>
> "Los llamamos insectos (bugs, y de aquí debugging) porque llamarlos errores sería psicológicamente inaceptable." —Hopkins
>
> "Las constantes no están, las variables no estarán." —Osborne

Depuración es el nombre que damos al proceso de ejecutar programas y cerciorarse de que los programas funcionan correctamente. "Correctamente" significa de acuerdo con el modelo mental que hemos construido o cómo el programa debe comportarse, sujeto a las ligaduras impuestas por el hardware. Aquí reside uno de los principales problemas; el operador y el hardware son los árbitros del rendimiento correcto. El hardware es generalmente implacable si hay un defecto en la forma en que se ha programado, lo mismo se hace exageradamente "no cooperativo" que no trabaja en absoluto. Por lo que respecta a la forma en que se ve el sistema, algunos controles muy sencillos, junto con intentos de utilizar el sistema para trabajos útiles durante unos días, harán que se domine el sistema bastante bien. Los problemas más difíciles serán los fallos intermitentes o las contradicciones lógicas.

Las computadoras son determinísticas. Es decir, si se empieza desde un estado conocido y se realiza una serie de operaciones conocidas, la computadora alcanzará siempre los mismos resultados. Alcanzar un estado conocido no es tan difícil —la puesta a cero del sistema y el borrado de la memoria lo harán—. Realizar una serie de operaciones conocidas no significa nada más que dejar correr otra vez el programa, aunque, si se están usando interrupciones, no se puede decir realmente que están siendo

realizadas exactamente las mismas operaciones, porque las interrupciones no ocurrirán *exactamente* al mismo tiempo que antes.

La aproximación "Orville Wright"

El papel del técnico al depurar un nuevo sistema CP/M es comparable a la popular, aunque falsa idea, de la forma en que los hermanos Wright desarrollaron sus máquinas voladoras: construir una máquina, llevarla a la cima de una colina, lanzarla y, cuando se rompe, examinar los restos para descubrir lo que estaba equivocado.

Cada vez que se realiza un ensamblaje y se comprueba, se está construyendo el avión y se le está echando a volar al borde de un acantilado. Cada vez que se estrella, se examina la chatarra y se intenta determinar la posible causa.

Este es un proceso muy inferencial. Con la chatarra como punto de partida, se utiliza la inferencia y la intuición para extrapolar el problema real y su corrección.

Programa de depuración interno

El concepto más importante que se puede necesitar al probar los sistemas CP/M es el mismo que se usa actualmente en las grabadoras de vuelo de las "cajas negras". Este mecanismo es esencialmente una grabadora de cinta de varios canales que graba todo de las condiciones relevantes del avión, su altura, altitud, agitación, etc., incluso las comunicaciones verbales entre los miembros de la tripulación. Si el aeroplano se estrella, los investigadores pueden reproducir la información y entender lo que ocurrió durante el vuelo.

Aplicando este concepto a la depuración del CP/M, significa que se debe construir en el programa algún método para grabar lo que está haciendo, de forma que si el sistema se estropea se pueda ver lo que estaba haciendo. Se puede lograr que el programa informe de lo que estaba equivocado.

El programa de depuración debería diseñarse al mismo tiempo que el resto del programa. Hay que planificar el programa de depuración cuando el diseño está aún en "sucio". El código fuente para la depuración debería ser una parte permanente del BIOS. Utilícese ensamblaje condicional con un "IF" de la mayor parte del programa de depuración para eliminarlo de la versión final, o hacer el programa sensible a un indicador en el bloque de configuración de forma que se pueda volver a hacer utilizable el programa de depuración en el momento de advertir que el sistema empieza a comportarse de forma extraña.

Cuanto más significativos son los datos de salida de la depuración,

menos habrá que adivinar qué es lo equivocado y, por consiguiente, el proceso será menos penoso y largo. La salida debe hacerse inteligible para otros que puedan usarla o para uno mismo al cabo de varios meses. Los datos que indican lo que está ocurriendo son más útiles que los valores hexadecimales internos, particularmente si alguien más ha de interpretarlos o transmitirlos telefónicamente.

Subrutinas de depuración

Muchos programadores realizan su depuración sobre una base de "coge lo que puedas coger", porque están sobrecargados de trabajo por la tarea de construir las herramientas necesarias. Otros son demasiado impacientes para poner en marcha un nuevo programa que les lleve unas cuantas horas o días para construir subrutinas de depuración.

Para ayudar a resolver este problema, la sección siguiente proporciona algunas herramientas prefabricadas que pueden utilizarse "tal cual". Cada una de estas rutinas se han depurado totalmente (¡no hay nada peor que un programa de desinsectación con insectos en él! —traducción literal—) y han sido utilizadas en la depuración de programas reales.

Filosofía general de diseño

Algunos métodos corrientes se muestran en los ejemplos que siguen. Estos incluyen la composición de "títulos" significativos (incluyendo la dirección específica que llamó la rutina de depuración), agrupando todos los programas de depuración, preservando los contenidos de todos los registros y preparando el área de pila de forma estándar.

Títulos del programa de depuración Cuando los contenidos de los registros o la memoria se envían como parte de un proceso de depuración, deberá componerse un título de texto explicativo describiendo los valores. Por ejemplo, mejor que visualizar el contenido del registro A como sigue,

```
A = 1F
```

se puede usar un título significativo tal como:

```
Transaction Code A = 1F.
```

Cuando se escribe un programa de depuración adicional, especialmente si se necesita añadirlo a una rutina existente, es incómodo tener que escribir la llamada a la rutina de depuración y buscar después a través del código

fuente para encontrar un lugar conveniente donde poner una hilera de título ASCII. Una hilera de título trasladada algunas páginas desde el punto en que está relacionada crea problemas cuando se quiere relacionar la salida en la pantalla o por impresora con el propio código fuente. Por consiguiente, todas las rutinas que siguen permiten declarar las hileras de títulos "en-línea" como la siguiente:

```
IF          DEBUG
CALL        Debug$Routine
DB          'Caption string here',CR,LF,0
ENDIF

MVI         .....           ;Instrucción siguiente
```

Todas las rutinas siguientes que envían un título reconocen un valor específico de 8 bits en la cadena del título. Si encuentran un valor 0ADH (nemónico por ADdress), enviarán la dirección del byte que sigue a la llamada a la rutina de depuración. Por ejemplo,

```
0210        CALL        Debug$Routine
0213        DB          0ADH,'Caption string',0
```

ocasionará que la rutina lance el siguiente mensaje:

```
0213 Caption string
```

Esto identifica el punto del programa desde el que se ha llamado a la rutina de depuración, y evita así cualquier posible ambigüedad entre llamadas distintas a la misma rutina con títulos similares.

Agrupación del programa de depuración La agrupación de todo el programa de depuración conduce al uso de un ensamblaje condicional con sentencias IF/ENDIF.

Fijación del área de pila Todas las rutinas que siguen preservan los registros de la CPU, de forma que no existen efectos secundarios cuando se usan. Todas ellas suponen que pueden utilizar el apuntador de pila y que hay suficiente sitio en esta área. Por tanto, se necesitará declarar el espacio de pila adecuado para el código principal y para las rutinas de depuración. Hay que llenar el área de pila con una configuración conocida, como la siguiente:

```
        DW   9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
        DW   9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
        DW   9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
Stack$Area:             ;Etiqueta borde superior del área
```

Entonces, durante la depuración, se puede examinar el área de pila y determinar cuánta queda sin utilizar. Por ejemplo, si se mirase el área de pila se vería algo así como:

```
                                    Indicador de "nivel"
                                                    V
99 99 99 99 99 99 99 99 99 99 99 99 09 15 43 42
01 29 00 00 1A 2B 10 FF FF 39 02 ED 11 01 37 44
DD 00 00 11 1A 23 31 00 41 AE FE 00 01 10 70 C9
```

Cuando se supera el área de pila pueden obtenerse errores misteriosos; el programa parece que salta en el espacio de forma no determinística. Al crear el área de pila de esta manera, se puede reconocer un desbordamiento fácilmente.

Inicialización de la depuración Antes de que se pueda ejecutar cualquiera de las subrutinas de depuración mencionadas en este capítulo, deberá hacerse una llamada a la subrutina de inicialización, DB$Init. La rutina DB$Init fija algunas de las variables internas que necesita el paquete de programas de depuración. Se puede necesitar añadir aquí algún trozo de programa para la propia inicialización.

Salida de la consola

Normalmente, se pueden utilizar las funciones CONOUT, tanto mediante el BDOS (función 2) como por medio del BIOS, llamando directamente al vector de saltos. Esto no puede hacerse cuando se necesite depurar las propias rutinas de consola, ni cuando se necesite depurar rutinas de interrupción. En el último caso, si una interrupción transfiere el control fuera de la rutina CONOUT en el BIOS, se tendrán reentradas no deseadas si el programa de depuración volvió a introducir el controlador de CONOUT para componer un título. Por consiguiente, las rutinas de depuración se han escrito para llamar su propia rutina CONOUT local, que se denomina DB$CONOUT. DB$CONOUT puede cambiarse para llamar al BDOS, al BIOS o a una rutina de salida "particular".

Se proporciona la rutina correspondiente DB$CONIN para entrada de consola, esencialmente por los mismos motivos.

Control de salida de depuración

Todas las salidas de las rutinas de depuración trabadas en este capítulo están controladas por un único indicador maestro, DB$Flag. Si es distinto de cero, se realizará la salida; si es cero, se suprimen todas las salidas.

Este indicador puede fijarse y borrarse desde cualquier parte del

programa que se está comprobando. Es especialmente útil cuando se necesita depurar una subrutina que es llamada muchas veces desde muchos sitios diferentes. Se pueden escribir trozos de programa adicionales para permitir la salida cuando se den ciertas condiciones; por ejemplo, cuando una pista o sector particular está a punto de ser escrito o cuando una memoria de entrada de caracteres está casi llena.

Las rutinas DB$On y DB$Off acceden al indicador de control mencionado y conectan y desconectan la salida, como su nombre sugiere.

La conexión y desconexión de la salida desde el interior del programa puede crear una imagen confusa de la salida de la depuración, debido a la falta de continuidad aparente. BD$Off da la opción de enviar una cadena de caracteres para indicar que la salida ha sido desconectada.

Contadores de pasos

Otro método de controlar la salida de la depuración es utilizar un *contador de pasos* que permita la salida sólo después de que el control haya pasado un número específico de veces a través de un punto particular del programa.

Se han previsto dos subrutinas para este propósito: DBSetPass pone el contador de pasos en un valor específico. DB$Pass decrementa esta cuenta de pasos cada vez que se le transfiere el control. Cuando el contador de pasos marca cero, el indicador de control DB$Flag no es cero y comienza la salida.

Utilizando las técnicas de contador de pasos se puede ahorrar tiempo y esfuerzo al seguir la pista a un problema que ocurre únicamente después de que el programa ha estado funcionando durante algunos minutos.

Visualización de contenidos de registros y memoria

La figura 10-2 muestra una serie de subrutinas de visualización, la primera de las cuales es DB$Display. Utiliza varios parámetros distintos, dependiendo de la información que se desee visualizar. La llamada genérica al DB$Display es la siguiente:

```
CALL      DB$Display
DB        Code      <- Indicada datos a
                       visualizar
{DW       Optional additional parameters}
 DB       'Caption string',0
```

Los distintos códigos que pueden utilizarse en esta llamada se muestran en la tabla 10-1.

Tabla 10-1. Códigos para DB$Display.

Código	Valor visualizado
Registros de 8 bits	
DB$F	Indicadores de condición
DB$A	Registro A
DB$B	Registro B
DB$C	Registro C
DB$D	Registro D
DB$E	Registro E
DB$H	Registro H
DB$L	Registro L
Memoria	
DB$Memory	Los bytes empiezan y acaban en las direcciones especificadas por los valores de dos palabras que siguen a la llamada
Registros de 16 bits	
DB$BC	Registro par BC
DB$DE	Registro par DE
DB$HL	Registro par HL
DB$SP	Indicador de pista
Valores de bytes	
DBBBC	Byte direccionado por BC
DBBDE	Byte direccionado por DE
DBBHL	Byte direccionado por HL
Valores de palabras	
DBWBC	Palabra direccionada por BC
DBWDE	Palabra direccionada por DE
DBWHL	Palabra direccionada por HL

La única función que utiliza parámetros adicionales es la DB$Memory. Esta permite la visualización de bytes de memoria en hexadecimal y ASCII, utilizando las direcciones de comienzo y final que siguen a la llamada, como se muestra en el ejemplo:

```
CALL      DB$Display
DB        DB$Memory
DW        Start$Address,End$Address
DB        'Caption string',0
```

Depuración de la lógica del programa

Además de visualizar los contenidos de los registros y de la memoria, se necesita tener acceso visual a los caminos de ejecución del programa, no en términos de direcciones, sino en términos de *problema*. Se puede hacer esto lanzando mensajes que indiquen las decisiones que se han tomado en el programa que se está ejecutando. Por ejemplo, si el BIOS controla un valor particular para ver si el sistema lee o escribe en un dispositivo particular, la rutina de depuración deberá componer un mensaje como éste:

```
Entering Disk Read Routine
```

Esto es más significativo que visualizar exclusivamente el código de función de los controladores —aunque pueda desearse componer éste igualmente en el caso de que se haya puesto a algún valor extraño.

Se suministran dos rutinas para componer mensajes de depuración. Estas son DB$MSG y DB$MSGI. Ambas componen cadenas de texto terminadas por un byte 00H. Se puede ver la diferencia entre dos subrutinas si se examina la forma de llamada para cada una de ellas.

DB$MSG se llama como sigue:

```
LXI  H,Message$Text      ;HL -> cadena de texto
CALL DB$MSG
```

DB$MSGI se llama como sigue:

```
CALL DB$MSG
DB   0DH,0AH,'Message Text',0 :En-línea
```

Resulta más conveniente el uso DB$MSGI. Si se decide que se necesita añadir un mensaje, se puede declarar el mensaje siguiendo inmediatamente a la llamada. Esto ayuda también cuando se mira el listado, porque se puede ver el texto completo de una ojeada.

Utilícese DB$MSG cuando el texto del mensaje necesite ser seleccionado de una tabla. Obténgase la dirección del texto en HL y llámese entonces al DB$MSG para lanzarlo.

Creación de mensajes visuales propios

Si se necesita construir rutinas de visualización especiales propias, puede resultar de ayuda el incorporar algunas de las pequeñas subrutinas del paquete de programas de depuración. Las rutinas que puede desearse utilizar son las siguientes:

DB$CONOUT
Visualiza el carácter que está en el registro C.

DB$CONIN
Devuelve el próximo carácter del teclado en A.

DB$CONINU
Devuelve el próximo carácter del teclado en A, convirtiendo las letras minúsculas en mayúsculas.

DB$DHLH
Visualiza el contenido de HL en hexadecimal.

DB$DAH
Visualiza el contenido de A en hexadecimal.

DB$CAH
Convierte el contenido de A en hexadecimal y lo almacena en la memoria apuntada por HL.

DB$Nibble$To$Hex
Convierte los cuatro bits menos significativos de A en un carácter hexadecimal en A.

DB$CRLF
Visualiza un CARRIAGE RETURN/LINE FEED.

DB$Colon
Visualiza la cadena " : ".

DB$Blank
Compone un carácter de un solo espacio.

DB$Flag$$ave$On
Guarda el estado actual del indicador de control de salida del depurador y entonces lo activa con el fin de permitir la salida.

DB$FlagSRestore
Vuelve a situar el indicador de control en el estado en que se encontraba cuando se llamó la última vez la rutina DB$Flag$Save$On.

DB$GHV
Obtiene un valor hexadecimal del teclado, lanzando antes un

mensaje de aviso. Pueden ser especificados de uno a cuatro caracteres como número máximo de caracteres a introducir.

DBATo$Upper
Si el registro A contiene una letra minúscula, la convierte en mayúscula.

Depuración de los controladores de entrada/salida

La depuración de los controladores de dispositivos de bajo nivel crea problemas especiales. El mayor de ellos es que normalmente no se desea leer y escribir por medio de los puertos hardware verdaderos mientras se está depurando el programa —tanto porque el hacerlo causaría la aparición de cosas extrañas en el hardware durante la depuración como porque se están desarrollando y depurando los controladores en un sistema diferente de la tarjeta hardware en la que los controladores tienen que trabajar.

Antes de tratar la solución, recuérdese que las instrucciones de entrada y salida (IN y OUT) tienen cada una dos bytes de longitud. El primer byte es el código de operación (0DBH para entrada, 0D3H para salida) y el segundo es el número de puerto para "entrar de" o "salir a".

Las subrutinas de los programas de depuración están previstas aquí para interceptar todas las instrucciones IN y OUT, visualizando el número de puerto y bien aceptando un valor hexadecimal de la consola y poniéndolo en el registro A (en el caso de IN) o bien visualizando los contenidos del registro A (para la instrucción OUT).

Las instrucciones IN y OUT pueden ser "atrapadas" cambiando el código de operación por una o dos instrucciones RST (nueva puesta en marcha). Un RST es efectivamente una instrucción CALL de un solo byte, llamando a una dirección previamente determinada en la memoria inferior. Las rutinas de depuración preparan a las instrucciones JMP de la memoria inferior para que reciban el control cuando se ejecuta un RST correcto. El programa que recibe el control puede elegir el número de puerto, visualizarlo y aceptar entonces un valor hexadecimal para el registro A (para IN) o visualizar los contenidos actuales del registro A (para OUT). Las subrutinas del ejemplo que se muestran más adelante en este capítulo utilizan RST 4 en lugar de instrucciones IN, y RST 5 para OUT.

Siempre que se plantee usar IN, habrá de utilizarse el programa siguiente:

```
IF        Debug
RST       4
ENDIF
IF        NOT Debug
DB        IN
ENDIF
DB        Port$Number
```

Obsérvese que se puede utilizar el código de operación IN como operando de una especificación DB. El ensamblador sustituye al código de operación correcto.

Utilícese el programa siguiente siempre que se necesite usar una instrucción OUT:

```
IF        Debug
RST       5
ENDIF
IF        NOT Debug
DB        OUT
ENDIF
DB        Port$Number
```

Cuando se ejecuta la instrucción RST 4 (IN), la subrutina de depuración visualiza

```
1AB3: Input from Port 01 : _
```

El "1AB3" es la dirección en memoria del byte que contiene el número de puerto. Sirve para apuntar exactamente la instrucción IN en memoria. Se puede introducir entonces uno o dos dígitos hexadecimales, que serán convertidos y se pondrán en el registro A antes de que el control vuelva al programa principal a la instrucción siguiente al byte que contiene el número de puerto.

Cuando la instrucción RST 5 (OUT) se encuentra, la subrutina de depuración visualiza

```
1AB5 :  Output to Port 01 : FF
```

Esto identifica dónde estaría normalmente la instrucción OUT, así como el número de puerto y los contenidos del registro A cuando el RST 5 (OUT) se ejecuta.

Depuración de rutinas del servicio de interrupciones

Puede utilizarse una técnica similar a la que describía en la instrucción RST para "falsear" una interrupción. Se fija previamente la dirección de la memoria inferior para la instrucción RST que se ha elegido para el salto en la rutina de servicio de interrupción que se está comprobando.

Cuando se ejecuta la instrucción RST, el control se transfiere a la rutina de servicio de interrupción exactamente como si hubiera ocurrido una interrupción. Sin duda se necesitará interceptar cualquier instrucción IN o OUT como se ha descrito anteriormente —de otra forma el programa iría probablemente a un bucle sin fin.

Antes de ejecutar la instrucción RST para falsear la interrupción, hay que cargar todos los registros con valores conocidos. Por ejemplo:

```
MVI     A,0AAH
LXI     B,0BBCCH
LXI     D,0DDEEH
LXI     H,01122H
RST     6       ;Simulación de interrupción
NOP
```

Cuando el control vuelve de la rutina de servicio, puede comprobarse si han realmacenado en todos los registros los valores correspondientes. Una rutina de servicio de interrupción que no restaure todos los registros puede producir perturbaciones muy difíciles de encontrar.

Compruébese también que el registro apuntador de pila ha sido almacenado de nuevo y que la rutina de servicio no requiere demasiados bytes de la pila.

Se puede usar también la instrucción CALL para transferir el control a la rutina de servicio de interrupción para falsear una interrupción. RST y CALL consiguen el mismo efecto, pero RST está más cerca de lo que ocurre cuando se realiza una interrupción real. Como es una instrucción de un solo byte, también es más fácil de modificar.

Listados de subrutinas

La figura 10-1 es un índice funcional del listado del código fuente de las subrutinas de depuración que se muestran en la figura 10-2. Los comentarios del listado definen precisamente cómo se hace una llamada a cada una de las subrutinas.

La figura 10-3 muestra la salida desde el banco de prueba de depuración.

Herramientas software para depuración

Además de la incorporación de subrutinas de depuración, puede necesitarse uno de los programas de depuración de programas particulares siguientes:

DDT (Herramienta dinámica de depuración)

Este programa, incluido en la edición estándar de CP/M, permite cargar programas, fijar y visualizar la memoria y los registros, rastrear el programa instrucción por instrucción o ejecutarlo a velocidad máxima, pero deteniéndose en ciertas direcciones (deno-

Línea de arranque	Componente funcional o rutinas
00001	Base de control de subrutina de depuración
00100	Composición de registro de control
00200	Composición de descarga control de memoria
00300	Composición para control de registro
00400	Composición indirecta del byte de control
00500	Control DB$On/Off
00600	Control DBSetPass y DB$Pass
00700	Entrada/salida control de desinsectación
00800	Subrutinas de desinsectación
01100	Inicialización DB$Init
01200	DB$CONINU—obtención de caracteres mayúsculos del teclado
01300	DB$CONIN—obtención de caracteres del teclado
01400	DB$CONOUT—composición carácter en C
01500	DB$On—posibilidad de salida desinsectación
01600	DB$Off—imposibilidad/no posibilidad salida desinsectación
01700	DBSetPas—fijación contador de pasos
01800	DB$Pass—ejecución contador de pasos
01900	DB$Display—rutina de composición principal de desinsectación
02200	Subrutinas de proceso de composición principal
02500	DB$Display$CALLA—composición de dirección de CALL
02600	DB$DHLH—composición HL en hexadecimal
02700	DB$DAH—composición A en hexadecimal
02800	DB$CAH—conversión de A en hexadecimal en memoria
02900	DB$Nibble$To$Hex—conversión de 4 bits LS de A en hexadecimal
02930	DB$CRLF—composición Carriage Return, Line Feed
02938	DB$Colon—composición ":"
02946	DB$Blank—composición " "
03100	DB$MSGI—composición mensaje en línea
03147	DB$MSG—composición mensaje dirigido por HL
03300	DB$Input—desinsectación rutina INput
03500	DB$Output—desinsectación rutina OUTput
03700	DB$Flag$Save$On—salvación de superficie y posibilidad de desinsectación
03800	DB$Flag$Restore—nuevo almacenamiento de la superficie de control de desinsectación
03900	DB$GHV—obtención de valor hexadecimal del teclado
04100	DBATo$Upper—conversión de A en caja mayúscula

Figura 10-1. Indice funcional para la figura 10-2.

minadas puntos de interrupción). Posee también un miniensamblador y un desensamblador incorporados de forma que no haya que ensamblar ninguno de los "añadidos" temporales que se hagan.

SID (Depuración simbólica interactiva)

De forma similar al DDT, el SID posee mejoras que son de ayuda si se utilizan MAC (macroensamblador) o RMAC (macroensamblador reubicado) de Digital Research. Ambos ensambladores pueden

```
00001
00002
00003                   ;
00004                   ;       Subrutinas de depuración
00005                   ;
00006                   ;<----  NOTA:
00007                   ;       Los números de línea que están en el margen izquierdo sirven
00008                   ;       únicamente para referenciar el programa en el texto.
00009                   ;       Existen discontinuidades deliberadas en los números
00010                   ;       de orden para permitir ampliaciones.
00011                   ;
00012                   ;       Debido a la necesidad de comprobar estas rutinas a fondo,
00013                   ;       y en caso de que se prevea realizar cambios, la rutina
00014                   ;       de base-de-comprobación para el paquete de programa
00015                   ;       de depuración ha sido suprimida de esta figura.
00016                   ;
00017                   ;       Depuración de la base de comprobación
00018                   ;
00019   0100                    ORG     100H
00020                   START:
00021   0100 316B03             LXI     SP,Test$Stack           ;Activar pila local
00022   0103 CDEA04             CALL    DB$Init                 ;Inicialización paquete de prog. de dep.
00023   0106 CD1505             CALL    DB$On                   ;Activación de salida de depuración
00024                                                           ;Comprobación simple visualización del reg. A
00025   0109 3EAA               MVI     A,0AAH                  ;Llenado de los demás reg. para comprobar la
00026   010B 01CCBB             LXI     B,0BBCCH                ;  visualización de la dep. y también para
00027   010E 11EEDD             LXI     D,0DDEEH                ;  comprobar el salvar/restituir registros
00028   0111 2111FF             LXI     H,0FF11H
00100                   ;#
00101                   ;       Comprobación visualización de registros
00102                   ;
00103   0114 B7                 ORA     A                       ;Puesta a 1 indicadores M y E, a 0 indicador Z
00104   0115 37                 STC                             ;Fijación acarreo
00105   0116 CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00106   0119 00                 DB      DB$F
00107   011A 466C616773         DB      'Flags',0
00108                   ;
00109   0120 CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00110   0123 02                 DB      DB$A
00111   0124 4120526567         DB      'A Register',0
00112                   ;
00113   012F CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00114   0132 04                 DB      DB$B
00115   0133 4220526567         DB      'B Register',0
00116                   ;
00117   013E CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00118   0141 06                 DB      DB$C
00119   0142 4320526567         DB      'C Register',0
00120                   ;
00121   014D CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00122   0150 08                 DB      DB$D
00123   0151 4420526567         DB      'D Register',0
00124                   ;
00125   015C CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00126   015F 0A                 DB      DB$E
00127   0160 4520526567         DB      'E Register',0
00128                   ;
00129   016B CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00130   016E 0C                 DB      DB$H
00131   016F 4820526567         DB      'H Register',0
00132                   ;
00133   017A CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00134   017D 0E                 DB      DB$L
00135   017E 4C20526567         DB      'L Register',0
00200                   ;#
00201                   ;       Comprobación visualización de volcado de memoria
00202                   ;
00203   0189 CD5205             CALL    DB$Display
00204   018C 18                 DB      DB$M                    ;Volcado de memoria
00205   018D 08012801           DW      108H,128H               ;Comprobación comienzo/final para no
00206   0191 4D656D6F72         DB      'Memory Dump #1',0      ;  múltiplos de 10H
00207                   ;
00208   01A0 CD5205             CALL    DB$Display
00209   01A3 18                 DB      DB$M                    ;Volcado de memoria
00210   01A4 00011F01           DW      100H,11FH               ;Comprobación comienzo/final sobre fronteras
00211   01A8 4D656D6F72         DB      'Memory Dump #2',0      ;  de línea visualizadas
00212                   ;
```

Figura 10-2. Subrutinas de depuración.

```
00213   01B7 CD5205             CALL    DB$Display
00214   01BA 18                 DB      DB$M                    ;Volcado de memoria
00215   01BB 01010001           DW      101H,100H               ;Comprobación manejo de errores donde
00216   01BF 4D656D6F72         DB      'Memory Dump #3',0      ;  dirección comienzo > dirección final
00217                   ;
00218   01CE CD5205             CALL    DB$Display
00219   01D1 18                 DB      DB$M                    ;Volcado de memoria
00220   01D2 00010001           DW      100H,100H               ;Comprobación último-caso de salida
00221   01D6 4D656D6F72         DB      'Memory Dump #4',0      ;  de byte simple
00300                   ;#
00301                   ;       Comprobación visualización de par de registros
00302                   ;
00303   01E5 CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00304   01E8 10                 DB      DB$BC
00305   01E9 4243205265         DB      'BC Register',0
00306                   ;
00307   01F5 CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00308   01F8 12                 DB      DB$DE
00309   01F9 4445205265         DB      'DE Register',0
00310                   ;
00311   0205 CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00312   0208 14                 DB      DB$HL
00313   0209 484C205265         DB      'HL Register',0
00314                   ;
00315   0215 CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00316   0218 16                 DB      DB$SP
00317   0219 5350205265         DB      'SP Register',0
00318                   ;
00319   0225 013203             LXI     B,Byte$BC               ;Fijar registros para comprobación de bytes
00320   0228 113303             LXI     D,Byte$DE
00321   022B 213403             LXI     H,Byte$HL
00400                   ;#
00401                   ;       Comprobación visualización indirecta de byte
00402                   ;
00403   022E CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00404   0231 1A                 DB      DB$B$BC
00405   0232 4279746520         DB      'Byte at (BC)',0
00406                   ;
00407   023F CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00408   0242 1C                 DB      DB$B$DE
00409   0243 4279746520         DB      'Byte at (DE)',0


00410                   ;
00411   0250 CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00412   0253 1E                 DB      DB$B$HL
00413   0254 4279746520         DB      'Byte at (HL)',0
00414                   ;
00415   0261 013503             LXI     B,Word$BC               ;Fijar registros para comprob. de palabra
00416   0264 113703             LXI     D,Word$DE
00417   0267 213903             LXI     H,Word$HL
00418
00419   026A CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00420   026D 20                 DB      DB$W$BC
00421   026E 576F726420         DB      'Word at (BC)',0
00422                   ;
00423   027B CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00424   027E 22                 DB      DB$W$DE
00425   027F 576F726420         DB      'Word at (DE)',0
00426                   ;
00427   028C CD5205             CALL    DB$Display              ;Llamada a rutina de depuración
00428   028F 24                 DB      DB$W$HL
00429   0290 576F726420         DB      'Word at (HL)',0
00500                   ;#
00501                   ;       Test DB$On/Off
00502                   ;
00503   029D CD1D05             CALL    DB$Off                  ;Desactivar salida de depuración
00504   02A0 CDD607             CALL    DB$MSGI                 ;Visualizar mensaje en-línea
00505   02A3 0D0A546869         DB      0DH,0AH,'This message should NOT appear',0
00506
00507   02C4 CD1505             CALL    DB$On
00508   02C7 CDD607             CALL    DB$MSGI
00509   02CA 0D0A446562         DB      0DH,0AH,'Debug output has been re-enabled.',0
00600                   ;#
00601                   ;       Comprobación lógica contador de pasos
00602                   ;
```

Figura 10-2. (Continuación.)

```
00603    02EE CD1D05            CALL    DB$Off                  ;Desactivar salida depuración
00604    02F1 CD2405            CALL    DB$Set$Pass             ;Fijar contador de pasos
00605    02F4 1E00              DW      30
00606                   ;
00607    02F6 3E22              MVI     A,34                    ;Fijar contador de bucle mayor que contador
00608                                                          ;  de pasos
00609                   Test$Pass$Loop:
00610    02F8 CD3505            CALL    DB$Pass                 ;Decrementar contador de pasos
00611    02FB CDD607            CALL    DB$MSGI                 ;Visualizar mensaje en-línea
00612    02FE 0D0A546869        DB      0DH,0AH,'This message should display 5 times',0
00613    0324 3D                DCR     A
00614    0325 C2F802            JNZ     Test$Pass$Loop
00700                   ;#
00701                   ;       Comprobación entrada/salida de depuración
00702                   ;
00703    0328 CD1D05            CALL    DB$Off                  ;Comprobación de que IN/OUT del programa de
00704                                                          ;  depuración aunque se haya
00705                                                          ;  desactivado la salida
00706    032B E7                RST     4                       ;Entrada de depuración
00707    032C 11                DB      11H                     ;Número de puerto
00708    032D EF                RST     5                       ;Sal. de dep. (retorno de valor desde la ent.)
00709    032E 22                DB      22H                     ;Número de puerto
00710
00711    032F C30000            JMP     0                       ;Arranque cal. al final de base-de-comprob.
00712                   ;
00713                   ;
00714                   ;       Valores ficticios para visualización de bytes y palabras
00715    0332 BC        Byte$BC:        DB      0BCH
00716    0333 DE        Byte$DE:        DB      0DEH
00717    0334 F1        Byte$HL:        DB      0F1H
00718                   ;
00719    0335 0C0B      Word$BC:        DW      0B0CH
00720    0337 0E0D      Word$DE:        DW      0D0EH
00721    0339 010F      Word$HL:        DW      0F01H
00722                   ;
00723    033B 9999999999                DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
00724    034B 9999999999                DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
00725    035B 9999999999                DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
00726                   Test$Stack:
00727                   ;
00728                   ;
00729                   ;
00730    0400                   ORG     400H                    ;Para prevenir listados innecesarios cuando
00731                                                          ;  cambia únicamente la base de comprob.
00732                   ;-------------------------------------------------------------------------
00800                   ;#
00801                   ;
00802                   ;       Subrutinas de depuración
00803                   ;
00804                   ;
00805                   ;       Equivalencias para códigos DB$Display
00806                   ;       Estas equivalencias son los desplazamientos en la tabla de
00807                   ;       direcciones para varias de las subrutinas que se utilizan.
00808                   ;
00809    0000 =         DB$F    EQU     00      ;Indicadores
00810    0002 =         DB$A    EQU     02      ;Registro A
00811    0004 =         DB$B    EQU     04      ;B
00812    0006 =         DB$C    EQU     06      ;C
00813    0008 =         DB$D    EQU     08      ;D
00814    000A =         DB$E    EQU     10      ;E
00815    000C =         DB$H    EQU     12      ;H
00816    000E =         DB$L    EQU     14      ;L
00817    0010 =         DB$BC   EQU     16      ;BC
00818    0012 =         DB$DE   EQU     18      ;DE
00819    0014 =         DB$HL   EQU     20      ;HL
00820    0016 =         DB$SP   EQU     22      ;Apuntador de pila
00821    0018 =         DB$M    EQU     24      ;Memoria
00822    001A =         DB$B$BC EQU     26      ;(BC)
00823    001C =         DB$B$DE EQU     28      ;(DE)
00824    001E =         DB$B$HL EQU     30      ;(HL)
00825    0020 =         DB$W$BC EQU     32      ;(BC+1),(BC)
00826    0022 =         DB$W$DE EQU     34      ;(DE+1),(DE)
00827    0024 =         DB$W$HL EQU     36      ;(HL+1),(HL)
00828                   ;
00829                   ;
00830                   ;       Equivalencias
00831    0020 =         RST4            EQU     20H     ;Dirección para RST 4 - Instrucción IN
```

Figura 10-2. (Continuación.)

```
00832    0028 =          RST5            EQU     28H     ;Dirección para RST 5 - Instrucción OUT
00833                    ;
00834    0001 =          B$CONIN         EQU     1       ;Código de la función CONIN del BDOS
00835    0002 =          B$CONOUT        EQU     2       ;Código de la función CONOUT del BDOS
00836    000A =          B$READCONS      EQU     10      ;Código de la función de lectura de consola del BDOS
00837    0005 =          BDOS            EQU     5       ;Punto de entrada al BDOS
00838                    ;
00839    0000 =          False           EQU     0
00840    FFFF =          True            EQU     NOT False
00841                    ;
00842                                                    ;Equivalencias para especificar cómo
00843                                                    ;  DB$CONOUT y DB$CONIN deberían
00844                                                    ;  realizar su entrada/salida
00845    0000 =          DB$Polled$IO    EQU     False   ;)
00846    0000 =          DB$BIOS$IO      EQU     False   ;) Sólo una es cierta
00847    FFFF =          DB$BDOS$IO      EQU     True    ;)
00848                    ;
00849                                                    ;Igualdades para entrada/salida
00850    0001 =          DB$Status$Port  EQU     01H     ;Puerto de estado de consola
00851    0002 =          DB$Data$Port    EQU     02H     ;Puerto de datos de consola
00852                    ;
00853    0002 =          DB$Input$Ready  EQU     0000$0010B      ;Datos para entrada, a punto
00854    0001 =          DB$Output$Ready EQU     0000$0001B      ;Preparado para salida
00855                    ;
00856                                                    ;Datos para entrada/salida del BIOS
00857    0400 C3         BIOS$CONIN:     DB      JMP     ;La rutina de inicialización fija estas
00858    0401 0000                       DW      0       ;  dos direcciones JMP en el BIOS
00859    0403 C3         BIOS$CONOUT:    DB      JMP
00860    0404 0000                       DW      0
00861                    ;
00862                    ;       Principales variables y constantes de depuración
00863                    ;
00864    0406 00         DB$Flag:        DB      0       ;Indicador principal de control de depuración
00865                                                    ;  Cuando este indicador es distinto de cero debe
00866                                                    ;  tener lugar toda la salida de depuración. Cuando
00867                                                    ;  es cero se suspende toda la salida de depuración.
00868                                                    ;  Puede ser modificado directamente por el
00869                                                    ;  usuario o utilizando las rutinas
00870                                                    ;  DB$On, DB$Off y DB$Pass.
00871                    ;
00872    0407 0000       DB$Pass$Count:  DW      0       ;Contador de pasos
00873                                                    ;  Cuando es distinto de cero la llamada a DB$Pass
00874                                                    ;  lo decrementa en uno. Cuando alcanza
00875                                                    ;  el valor cero el indicador de control
00876                                                    ;  de depuración DB$Flag es puesto a cero con
00877                                                    ;  lo que se activa la salida de depuración.
00878                    ;
00879                    DB$Save$HL:                    ;Almacenar área para HL
00880    0409 00         DB$Save$L:      DB      0
00881    040A 00         DB$Save$H:      DB      0
00882
00883    040B 0000       DB$Save$SP:     DW      0       ;Salvar área para apuntador de pila
00884    040D 0000       DB$Save$RA:     DW      0       ;Almacenar área para dirección de retorno
00885    040F 0000       DB$Call$Address:        DW      0       ;Comienza lo mismo que DB$Save$RA,
00886                                                    ;  pero DB$Save$RA se actualiza durante
00887                                                    ;  el proceso de depuración. El valor se lanza
00888                                                    ;  a la salida antes del título
00889                    DB$Start$Address:              ;Dirección de comienzo para visualización de memoria
00890    0411 0000                       DW      0
00891                    DB$End$Address:                ;Dirección final para visualización de memoria
00892    0413 0000                       DW      0
00893                    DB$Display$Code:               ;Visualización del código requerido
00894    0415 00                         DB      0
00895                    ;
00896                    ;
00897                                                    ;Area de pila
00898    0416 9999999999                 DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
00899    0426 9999999999                 DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
00900    0436 9999999999                 DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
00901    0446 00         DB$Save$E:      DB      0       ;Registro E
00902    0447 00         DB$Save$D:      DB      0       ;Registro D
00903    C448 00         DB$Save$C:      DB      0       ;Registro C
00904    0449 00         DB$Save$B:      DB      0       ;Registro B
00905    044A 00         DB$Save$F:      DB      0       ;Indicadores
00906    044B 00         DB$Save$A:      DB      0       ;Registro A
00907                    DB$Stack:                      ;Depuración de área de pila
00908                                                    ;  Los reg. en el área de pila son "introducidos"
00909                                                    ;  (PUSHed) en la pila y accedidos directamente.
```

Figura 10-2. (Continuación.)

```
00910                   ;
00911                   ;       Registro de mensajes de título
00912                   ;
00913                   ;       La tabla siguiente, indexada por Display$Code, se utiliza para acceder al
00914                   ;       registro de cadena de título
00915                   ;
00916                   DB$Register$Captions:
00917   044C 7204               DW      DB$F$RC         ;Indicadores
00918   044E 7804               DW      DB$A$RC         ;Registro A
00919   0450 7A04               DW      DB$B$RC         ;B
00920   0452 7C04               DW      DB$C$RC         ;C
00921   0454 7E04               DW      DB$D$RC         ;D
00922   0456 8004               DW      DB$E$RC         ;E
00923   0458 8204               DW      DB$H$RC         ;H
00924   045A 8404               DW      DB$L$RC         ;L
00925   045C 8604               DW      DB$BC$RC        ;BC
00926   045E 8904               DW      DB$DE$RC        ;DE
00927   0460 8C04               DW      DB$HL$RC        ;HL
00928   0462 8F04               DW      DB$SP$RC        ;Apuntador de pila
00929   0464 9204               DW      DB$M$RC         ;Memoria
00930   0466 A604               DW      DB$B$BC$RC      ;(BC)
00931   0468 AB04               DW      DB$B$DE$RC      ;(DE)
00932   046A B004               DW      DB$B$HL$RC      ;(HL)
00933   046C B504               DW      DB$W$BC$RC      ;(BC+1),(BC)
00934   046E C104               DW      DB$W$DE$RC      ;(DE+1),(DE)
00935   0470 CD04               DW      DB$W$HL$RC      ;(HL+1),(HL)
00936                   ;
00937   0472 466C616773DB$F$RC:         DB      'Flags',0       ;Indicadores
00938   0478 4100       DB$A$RC:        DB      'A',0           ;Registro A
00939   047A 4200       DB$B$RC:        DB      'B',0           ;B
00940   047C 4300       DB$C$RC:        DB      'C',0           ;C
00941   047E 4400       DB$D$RC:        DB      'D',0           ;D
00942   0480 4500       DB$E$RC:        DB      'E',0           ;E
00943   0482 4800       DB$H$RC:        DB      'H',0           ;H
00944   0484 4C00       DB$L$RC:        DB      'L',0           ;L
00945   0486 424300     DB$BC$RC:       DB      'BC',0          ;BC
00946   0489 444500     DB$DE$RC:       DB      'DE',0          ;DE
00947   048C 484C00     DB$HL$RC:       DB      'HL',0          ;HL
00948   048F 535000     DB$SP$RC:       DB      'SP',0          ;Apuntador de pila
00949   0492 5374617274DB$M$RC:         DB      'Start, End Address ',0 ;Memoria
00950   04A6 2842432900DB$B$BC$RC:      DB      '(BC)',0        ;(BC)
00951   04AB 2844452900DB$B$DE$RC:      DB      '(DE)',0        ;(DE)
00952   04B0 28484C2900DB$B$HL$RC:      DB      '(HL)',0        ;(HL)
00953   04B5 2842432B31DB$W$BC$RC:      DB      '(BC+1),(BC)',0 ;(BC+1),(BC)
00954   04C1 2844452B31DB$W$DE$RC:      DB      '(DE+1),(DE)',0 ;(DE+1),(DE)
00955   04CD 28484C2B31DB$W$HL$RC:      DB      '(HL+1),(HL)',0 ;(HL+1),(HL)
00956                   ;
00957                   ;       Indicadores de mensaje
00958                   ;
00959   04D9 43785A784DDB$Flags$Msg:    DB      'CxZxMxExIx',0  ;Compatible con la visualización del DDT
00960                   ;
00961                   ;       Máscara de indicadores utilizada para comprobar byte indicador de usuario
00962                   ;
00963                   DB$Flag$Masks:
00964   04E4 01                         DB      0000$0001B      ;Acarreo
00965   04E5 40                         DB      0100$0000B      ;Cero
00966   04E6 80                         DB      1000$0000B      ;Menos
00967   04E7 04                         DB      0000$0100B      ;Paridad par
00968   04E8 10                         DB      0001$0000B      ;Acarreo interdígito (acarreo aux.)
00969   04E9 00                         DB      0               ;Terminal
01100                   ;*
01101                   ;       DB$Init
01102                   ;       Esta rutina inicializa el paquete de programas de depuración
01103                   ;
01104                   DB$Init:
01105                           IF      DB$BIOS$IO              ;Utilización del BIOS para CONIN/CONOUT
01106                           LHLD    1                       ;Obt. de direc. de arranque cal. desde pág.
01107                                                           ;  base. H = pág. del vec. de sal. del BIOS
01108                           MVI     L,09H                   ;Obt. del despl. de CONIN en vec. de saltos
01109                           SHLD    BIOS$CONIN + 1          ;Fijar dirección
01110                           MVI     L,0CH                   ;Obt. del despl. CONOUT en vector
01111                           SHLD    BIOS$CONOUT + 1         ;   de saltos
01112                           ENDIF
01113
01114                                                   ;Fijar instrucciones JMP para recibir el control
01115                                                   ;  cuando se ejecute una instrucción RST
01116   04EA 3EC3               MVI     A,JMP                   ;Fijar instrucciones JMP en los puntos RST
```

Figura 10-2. (Continuación.)

```
01117    04EC 322000            STA     RST4
01118    04EF 322800            STA     RST5
01119    04F2 211A08            LXI     H,DB$Input      ;Dirección de la rutina de entrada falseada
01120    04F5 222100            SHLD    RST4 + 1
01121    04F8 216C08            LXI     H,DB$Output     ;Dirección de la rutina de salida falseada
01122    04FB 222900            SHLD    RST5 + 1
01123
01124    04FE C9                RET
01200                   ;#
01201                   ;       DB$CONINU
01202                   ;       Esta rutina devuelve el siguiente carácter de la consola,
01203                   ;       pero convirtiéndolo de la "a" a la "z" a mayúsculas.
01204                   ;
01205                   DB$CONINU:
01206    04FF CD0505            CALL    DB$CONIN                ;Obtención de carácter del teclado
01207    0502 C31B09            JMP     DB$A$To$Upper           ;Cambio a mayúsculas y retorno
01300                   ;#
01301                   ;       DB$CONIN
01302                   ;       Esta rutina devuelve el carácter siguiente de la consola.
01303                   ;       De acuerdo con la configuración de equivalencias utiliza
01304                   ;       entrada/salida por bytes del BDOS (función 2) o el BIOS.
01305                   ;
01306                   ;       Parámetros de salida
01307                   ;
01308                   ;               A = carácter desde consola
01309                   ;
01310                   DB$CONIN:
01311                           IF      DB$Polled$IO            ;Simple entrada
01312                           IN      DB$Status$Port          ;Comprobación de datos entrando
01313                           ANI     DB$Input$Ready
01314                           JZ      DB$CONIN                ;No
01315                           IN      DB$Data$Port            ;Entrada de carácter de datos
01316                           PUSH    PSW                     ;Almacenar carácter-dato
01317                           MOV     C,A                     ;Preparado para salida
01318                           CALL    DB$CONOUT               ;Repetición
01319                           POP     PSW                     ;Recuperación carácter-dato
01320                           RET
01321                           ENDIF
01322
01323                           IF      DB$BDOS$IO              ;Utilización del BDOS para entrada
01324    0505 0E01              MVI     C,B$CONIN               ;Lectura de consola
01325    0507 C30500            JMP     BDOS                    ;El BDOS retorna a su llamador
01326                           ENDIF
01327
01328                           IF      DB$BIOS$IO              ;Utilización del BIOS para entrada
01329                           JMP     BIOS$CONIN              ;Se fija durante la inicialización
01330                                                           ;  del BIOS
01331                           ENDIF
01332
01400                   ;#
01401                   ;       DB$CONOUT
01402                   ;       Esta rutina lanza a salida por consola el carácter que se encuentra
01403                   ;       en el registro utilizando entrada salida simple el BDOS o el BIOS.
01404                   ;
01405                   ;       Parámetros de entrada
01406                   ;               A = byte a lanzar a salida
01407                   ;
01408                   DB$CONOUT:
01409    050A 3A0604            LDA     DB$Flag                 ;Comprobación de si está activa salida dep.
01410    050D B7                ORA     A
01411    050E C8                RZ                              ;Ignorar la salida si desactivado
01412
01413                           IF      DB$Polled$IO            ;Utilización salida simple
01414                           IN      DB$Status$Port          ;Comprobación de preparado para salida
01415                           ANI     DB$Output$Ready
01416                           JZ      DB$CONOUT               ;No
01417                           MOV     A,C                     ;Obtención de byte de datos
01418                           OUT     DB$Data$Port
01419                           RET
01420                           ENDIF
01421
01422                           IF      DB$BDOS$IO              ;Utilización del BDOS para salida
01423    050F 59                MOV     E,C                     ;Traslado al registro correcto
01424    0510 0E02              MVI     C,B$CONOUT
01425    0512 C30500            JMP     BDOS                    ;Retorno del BDOS a quien le llamó
01426                           ENDIF
01427
01428                           IF      DB$BIOS$IO              ;Utilización del BIOS para salida
```

Figura 10-2. (Continuación.)

```
01429                                   MOV     A,C                     ;Trasladar a registro correcto
01430                                   JMP     BIOS$CONOUT             ;Fijación durante la inicialización de la
01431                                   ENDIF                           ;  depuración
01500                           ;#
01501                           ;
01502                           ;       DB$On
01503                           ;       Esta rutina activa todas las salidas de depuración fijando
01504                           ;       DB$Flag a distinto de cero.
01505                           ;
01506                           DB$On:
01507   0515 F5                         PUSH    PSW                     ;Preservar registros
01508   0516 3EFF                       MVI     A,0FFH
01509   0518 320604                     STA     DB$Flag                 ;Fijar a uno indicador de control
01510   051B F1                         POP     PSW
01511   051C C9                         RET
01600                           ;#
01601                           ;
01602                           ;       DB$Off
01603                           ;       Esta rutina desactiva todas las salidas de la depuración
01604                           ;       fijando a cero DB$Flag.
01605                           ;
01606                           DB$Off:
01607   051D F5                         PUSH    PSW                     ;Preservar registros
01608   051E AF                         XRA     A
01609   051F 320604                     STA     DB$Flag                 ;Puesta a cero del indicador de control
01610   0522 F1                         POP     PSW
01611   0523 C9                         RET
01700                           ;#
01701                           ;
01702                           ;       DB$Set$Pass
01703                           ;       Esta rutina fija el contador de pasos. Las llamadas siguientes
01704                           ;       a DB$Pass decrementarán el contador y cuando se alcance
01705                           ;       el 0 se activa la salida de depuración.
01706                           ;
01707                           ;       Secuencia de llamada
01708                           ;
01709                           ;               CALL    DB$Set$Pass
01710                           ;               DW      Pass$Count$Value
01711                           ;
01712                           DB$Set$Pass:
01713   0524 220904                     SHLD    DB$Save$HL              ;Preservar HL de usuario
01714   0527 E1                         POP     H                       ;Recuperación dirección de retorno
01715   0528 D5                         PUSH    D                       ;Preservar DE de usuario
01716   0529 5E                         MOV     E,M                     ;Obtención byte LS
01717   052A 23                         INX     H                       ;Actualización apuntador
01718   052B 56                         MOV     D,M                     ;Obtención byte MS
01719   052C 23                         INX     H                       ;HL apunta a la dirección de retorno
01720   052D EB                         XCHG                            ;HL = contador de pasos
01721   052E 220704                     SHLD    DB$Pass$Count           ;Fijación contador de pasos de depuración
01722   0531 EB                         XCHG                            ;HL apunta a la dirección de retorno
01723   0532 D1                         POP     D                       ;Recuperación de DE del usuario
01724   0533 E3                         XTHL                            ;Recuperación HL del usuario y fijación de
01725                                                                   ;  la direc. de retorno en cabecera de pila
01726   0534 C9                         RET
01800                           ;#
01801                           ;
01802                           ;       DB$Pass
01803                           ;       Esta rutina decrementa el contador de pasos de depuración -
01804                           ;       si el resultado es negativo, no se toma ninguna acción.
01805                           ;       Si el resultado es cero, fija a uno el indicador de control de
01806                           ;       depuración para activar la salida.
01807                           ;
01808                           DB$Pass:
01809   0535 F5                         PUSH    PSW                     ;Almacenar registros de usuario
01810   0536 E5                         PUSH    H
01811   0537 2A0704                     LHLD    DB$Pass$Count           ;Obtención contador de pasos
01812   053A 2B                         DCX     H
01813   053B 7C                         MOV     A,H                     ;Comprobación de conteo negativo
01814   053C B7                         ORA     A
01815   053D FA4705                     JM      DB$Pass$x               ;Si, no tomar ninguna acción adicional
01816   0540 220704                     SHLD    DB$Pass$Count           ;Almacenar contador decrementado
01817   0543 B5                         ORA     L                       ;Comprobación de contador cero
01818   0544 CA4A05                     JZ      DB$Pass$ED              ;Si, activar depuración
01819                           DB$Pass$x:
01820   0547 E1                         POP     H                       ;Recuperación registros de usuario
01821   0548 F1                         POP     PSW
01822   0549 C9                         RET
```

Figura 10-2. (Continuación.)

```
01823                           ;
01824                           DB$Pass$Ed:                             ;Activación depuración
01825   054A 3EFF               MVI     A,0FFH
01826   054C 320604             STA     DB$Flag                         ;Fijación indicador de control depuración
01827   054F C34705             JMP     DB$Pass$x
01900                           ;#
01901                           ;
01902                           ;       DB$Display
01903                           ;       Esta es la rutina primaria de visulización de depuración
01904                           ;
01905                           ;       Secuencia de llamada
01906                           ;
01907                           ;               CALL    DB$Display
01908                           ;               DB      Display$Code
01909                           ;               DB      'Caption String',0
01910                           ;
01911                           ;               El código de visualización (display) identifica qué registro(s)
01912                           ;               debe ser visualizado.
01913                           ;
01914                           ;               Cuando el programa de visualización especifica un bloque
01915                           ;               de memoria la secuencia es:
01916                           ;
01917                           ;               CALL    DB$Display
01918                           ;               DB      Display$Code
01919                           ;               DW      Start$Address,End$Address
01920                           ;               DB      'Caption String',0
01921                           ;
01922                           DB$Display:
01923                           ;
01924                           DB$Display$Enabled:
01925   0552 220904             SHLD    DB$Save$HL                      ;Salvar HL de usuario
01926
01927   0555 E3                 XTHL                                    ;Obtención dirección de retorno en la pila
01928   0556 220D04             SHLD    DB$Save$RA                      ;Esta obtención está act. por el pros. de dep.
01929   0559 E5                 PUSH    H                               ;Almacenar la direc. de retorno temporalmente
01930   055A 2B                 DCX     H                               ;Restar 3 a la dirección de la instrucción
01931   055B 2B                 DCX     H                               ;  de llamada
01932   055C 2B                 DCX     H
01933   055D 220F04             SHLD    DB$Call$Address                 ;Almacenar dirección real de llamada
01934   0560 E1                 POP     H                               ;Recuperar dirección de retorno
01935
01936   0561 F5                 PUSH    PSW                             ;Almacenamiento temporal de los indicadores
01937                                                                   ;  para prevenir que sean cambiados por DAD SP
01938   0562 210000             LXI     H,0                             ;Preservar ajuntador de pila
01939   0565 39                 DAD     SP
01940   0566 23                 INX     H                               ;Correcto para PUSH PSW extra necesario para
01941   0567 23                 INX     H                               ;  salvar indicadores
01942   0568 220B04             SHLD    DB$Save$SP
01943   056B F1                 POP     PSW                             ;Recuperar indicadores
01944
01945   056C 314C04             LXI     SP,DB$Stack                     ;Conmutar a pila local
01946
01947   056F F5                 PUSH    PSW                             ;Almacenar otros registros de usuario
01948   0570 C5                 PUSH    B                               ;El área de pila se ha diseñado especialmente
01949   0571 D5                 PUSH    D                               ;  para acceder a estos registros
01950
01951   0572 2A0D04             LHLD    DB$Save$RA                      ;Obtención dirección de retorno
01952   0575 7E                 MOV     A,M                             ;Obtención código de visualización
01953   0576 321504             STA     DB$Display$Code
01954   0579 23                 INX     H                               ;Actualización dirección de retorno
01955
01956   057A FE18               CPI     DB$M                            ;Comprobación de si debe visualizarse la
01957   057C C29105             JNZ     DB$Not$Memory                   ;  memoria
01958   057F 5E                 MOV     E,M                             ;Obtención DE = dirección comienzo
01959   0580 23                 INX     H
01960   0581 56                 MOV     D,M
01961   0582 23                 INX     H
01962   0583 EB                 XCHG                                    ;HL = dirección de comienzo
01963   0584 221104             SHLD    DB$Start$Address
01964   0587 EB                 XCHG                                    ;HL -> dirección de final
01965   0588 5E                 MOV     E,M                             ;Obtención de DE = dirección de final
01966   0589 23                 INX     H
01967   058A 56                 MOV     D,M
01968   058B 23                 INX     H
01969   058C EB                 XCHG                                    ;HL = dirección de final, DE -> título
01970   058D 221304             SHLD    DB$End$Address
01971   0590 EB                 XCHG                                    ;HL -> cadena título
```

Figura 10-2. (Continuación.)

```
01972                   DB$Not$Memory:
01973                   ;
01974                   ;       Salida de cadena de preámbulo y de titulo
01975                   ;       El formato para cualquier visualización excepto de la memoria es:
01976                   ;
01977                   ;       nnnn : Cadena titulo : RC = vvvv
01978                   ;        ^                     ^    ^
01979                   ;       Dirección de llamada  :    Valor
01980                   ;                       Titulo registro (A,B,C...)
01981                   ;
01982                   ;       Se lanza un retorno de carro, salto de linea al comienzo del
01983                   ;       mensaje, no al finalizar.
01984                   ;
01985                   ;       La visualización de memoria es como sigue:
01986                   ;
01987                   ; nnnn : Cadena titulo : Comienzo, Final : ssss, eeee
01988                   ; ssss : hh hh hh hh hh hh hh hh hh hh hh hh hh hh hh hh : cccc cccc cccc cccc
01989                   ;
01990
01991   0591 E5                 PUSH    H                       ;Almacenar apuntador a cadena titulo
01992   0592 CDC107             CALL    DB$CRLF                 ;Visualización ret. de carro, salto de linea
01993   0595 CD7C07             CALL    DB$Display$CALLA        ;Visualización DB$Call$Address en hex.
01994
01995   0598 E1                 POP     H                       ;Recuperar apuntador a cadena titulo
01996                   DB$Display$Caption:                     ;HL -> cadena titulo
01997   0599 7E                 MOV     A,M                     ;Obtención de carácter
01998   059A 23                 INX     H
01999   059B B7                 ORA     A                       ;Comprobación de final de cadena
02000   059C CAA805             JZ      DB$End$Caption          ;Si
02001
02002   059F E5                 PUSH    H                       ;Almacenar apuntador a cadena
02003   05A0 4F                 MOV     C,A                     ;Preparado para salida
02004   05A1 CD0A05             CALL    DB$CONOUT               ;Visualización de carácter
02005   05A4 E1                 POP     H                       ;Recuperación de apuntador a cadena
02006   05A5 C39905             JMP     DB$Display$Caption      ;Vuelta atrás a por nuevo carácter
02007                   ;
02008                   DB$End$Caption:
02009   05A8 220D04             SHLD    DB$Save$RA              ;Almacenamiento direc. de retorno actualizada
02010
02011   05AB CDC807             CALL    DB$Colon                ;Visualización de ' : '
02012
02013                                                           ;Visualización de titulo registro
02014   05AE 3A1504             LDA     DB$Display$Code         ;Obtención cód. de visualización de usuario
02015   05B1 5F                 MOV     E,A                     ;Componer código de visualización en palabra
02016   05B2 1600               MVI     D,0
02017   05B4 D5                 PUSH    D                       ;Salvar valor de palabra para utiliz. post.
02018
02019   05B5 FE18               CPI     DB$M                    ;Visualización de mem. es un caso especial
02020   05B7 CACF05             JZ      DB$Display$Mem$Caption  ;Si
02021
02022   05BA 214C04             LXI     H,DB$Register$Captions  ;Componer apuntador a dirección en tabla
02023   05BD 19                 DAD     D                       ;HL -> palabra conteniendo dirección
02024                                                           ;  de titulo de registro
02025   05BE 5E                 MOV     E,M                     ;Obtención byte LS de dirección
02026   05BF 23                 INX     H
02027   05C0 56                 MOV     D,M                     ;DE -> cadena de titulo de registro
02028   05C1 EB                 XCHG                            ;HL -> cadena de titulo de registro
02029   05C2 CDEE07             CALL    DB$MSG                  ;Visualización de mensaje direc. por HL
02030
02031   05C5 CDD607             CALL    DB$MSGI                 ;Visualización de mensaje en-linea
02032   05C8 203D2000           DB      ' = ',0
02033   05CC C3ED05             JMP     DB$Select$Routine       ;Ir a procesador correcto
02034                   ;
02035                   DB$Display$Mem$Caption:                 ;La visualización de memoria necesita un
02036                                                           ;  titulo especial con las direcciones
02037                                                           ;  de comienzo y final
02038   05CF 219204             LXI     H,DB$M$RC               ;Visualización titulo especifico
02039   05D2 CDEE07             CALL    DB$MSG
02040   05D5 CDC807             CALL    DB$Colon                ;Visualización ' : '
02041
02042   05D8 2A1104             LHLD    DB$Start$Address        ;Visualización dirección de comienzo
02043   05DB CD8707             CALL    DB$DHLH                 ;Visualización de HL en hexadecimal
02044
02045   05DE CDD607             CALL    DB$MSGI                 ;Visualización de mensaje en-linea
02046   05E1 2C2000             DB      ', ',0
02047
02048   05E4 2A1304             LHLD    DB$End$Address          ;Obtención dirección de finalización
```

Figura 10-2. (Continuación.)

```
02049    05E7 CD8707          CALL    DB$DHLH                 ;Visualización de HL en hexadecimal
02050    05EA CDC107          CALL    DB$CRLF                 ;Visualización de ret. de carro, sal. de línea
02051                                                         ;Salto a rutina seleccionada
02052                 DB$Select$Routine:
02053    05ED D1              POP     D                       ;Recuperación palabra-valor Display$Code
02054    05EE 210A06          LXI     H,DB$Display$Table
02055    05F1 19              DAD     D                       ;HL -> dirección del código para procesar
02056                                                         ;  requerimientos de la visualización
02057    05F2 5E              MOV     E,M                     ;Obtención byte LS de dirección
02058    05F3 23              INX     H                       ;Actualización de apuntador
02059    05F4 56              MOV     D,M                     ;Obtención byte MS de dirección
02060    05F5 EB              XCHG                            ;HL -> código
02061
02062    05F6 11FB05          LXI     D,DB$Exit               ;Falsear enlace en la pila
02063    05F9 D5              PUSH    D
02064    05FA E9              PCHL                            ;Llamada al procesador de visualización
02065                 ;
02066                 DB$Exit:                                ;Retorno al usuario
02067    05FB D1              POP     D                       ;Recuperación registros de usuario
02068    05FC C1              POP     B                       ;  almacenados en pila de depuración local
02069    05FD F1              POP     PSW
02070    05FE 2A0B04          LHLD    DB$Save$SP              ;Vuelta a la pila de usuario
02071    0601 F9              SPHL
02072    0602 2A0D04          LHLD    DB$Save$RA              ;Obtención de dirección de retorno
02073                                                         ;  actualizada (ignora parámetros en-línea)
02074    0605 E3              XTHL                            ;Reemplazar en cabecera de pila de usuario
02075    0606 2A0904          LHLD    DB$Save$HL              ;Obtención HL de usuario
02076    0609 C9              RET                             ;Transferencia a dirección de retorno
02077                                                         ;  correcta
02078
02079                 DB$Display$Table:
02080    060A 3006            DW      DP$F                    ;Indicadores
02081    060C 5406            DW      DP$A                    ;Registro A
02082    060E 5A06            DW      DP$B                    ;B
02083    0610 6006            DW      DP$C                    ;C
02084    0612 6606            DW      DP$D                    ;D
02085    0614 6C06            DW      DP$E                    ;E
02086    0616 7206            DW      DP$H                    ;H
02087    0618 7806            DW      DP$L                    ;L
02088    061A 7E06            DW      DP$BC                   ;BC
02089    061C 8406            DW      DP$DE                   ;DE
02090    061E 8A06            DW      DP$HL                   ;HL
02091    0620 9006            DW      DP$SP                   ;Apuntador de pila
02092    0622 9606            DW      DP$M                    ;Memoria
02093    0624 4907            DW      DP$B$BC                 ;(BC)
02094    0626 5007            DW      DP$B$DE                 ;(DE)
02095    0628 5707            DW      DP$B$HL                 ;(HL)
02096    062A 5E07            DW      DP$W$BC                 ;(BC+1),(BC)
02097    062C 6807            DW      DP$W$DE                 ;(DE+1),(DE)
02098    062E 7207            DW      DP$W$HL                 ;(HL+1),(HL)
02200                 ;#
02201                 ;       Rutinas de procesamiento de la visualización de la depuración
02202                 ;
02203                 DP$F:                                   ;Indicadores
02204                                                         ;Los indicadores se visualizan en la misma forma
02205                                                         ;  que el DDT utiliza: C1Z0M0B0I0
02206    0630 3A4A04          LDA     DB$Save$F               ;Obtención de indicadores
02207    0633 47              MOV     B,A                     ;Preservar copia
02208    0634 21DA04          LXI     H,DB$Flags$Msg + 1      ;HL -> primer 0/1 en mensaje
02209    0637 11E404          LXI     D,DB$Flag$Masks         ;DE -> tabla de valores de máscara de indicadores
02210                 DB$F$Next:
02211    063A 1A              LDAX    D                       ;Obtención máscara de indicadores siguiente
02212    063B B7              ORA     A                       ;Comprobación de final de tabla
02213    063C CA4E06          JZ      DB$F$Display            ;Sí, visualizar resultados
02214
02215    063F A0              ANA     B                       ;Comprobación de indicador a uno
02216    0640 3E31            MVI     A,'1'                   ;Suposición de que sí lo está
02217    0642 C24706          JNZ     DB$F$NZ                 ;Sí, está a uno
02218    0645 3E30            MVI     A,'0'                   ;No, está a cero
02219                 DB$F$NZ:
02220    0647 77              MOV     M,A                     ;Almacenar '0' o '1' en texto del mensaje
02221    0648 23              INX     H                       ;Actualización apuntador al próximo 0/1
02222    0649 23              INX     H
02223    064A 13              INX     D                       ;Actualización apuntador a máscara de indicadores
02224    064B C33A06          JMP     DB$F$Next
02225                 DB$F$Display:                           ;Visualización de resultados
02226    064E 21D904          LXI     H,DB$Flags$Msg
```

Figura 10-2. (Continuación.)

```
02227   0651 C3EE07             JMP     DB$MSG          ;Visualización de mensaje y retorno
02228                   ;
02229                   DP$A:           ;A register
02230   0654 3A4B04             LDA     DB$Save$A       ;Obtención de valor almacenado
02231   0657 C39107             JMP     DB$DAH          ;Visualización de éste y retorno
02232                   ;
02233                   DP$B:           ;B
02234   065A 3A4904             LDA     DB$Save$B       ;Obtención de valor almacenado
02235   065D C39107             JMP     DB$DAH          ;Visualización de éste y retorno
02236                   ;
02237                   DP$C:           ;C
02238   0660 3A4B04             LDA     DB$Save$C       ;Obtención de valor almacenado
02239   0663 C39107             JMP     DB$DAH          ;Visualización de éste y retorno
02240                   ;
02241                   DP$D:           ;D
02242   0666 3A4704             LDA     DB$Save$D       ;Obtención de valor almacenado
02243   0669 C39107             JMP     DB$DAH          ;Visualización de éste y retorno
02244                   ;
02245                   DP$E:           ;E
02246   066C 3A4604             LDA     DB$Save$E       ;Obtención de valor almacenado
02247   066F C39107             JMP     DB$DAH          ;Visualización de éste y retorno
02248                   ;
02249                   DP$H:           ;H
02250   0672 3A0A04             LDA     DB$Save$H       ;Obtención de valor almacenado
02251   0675 C39107             JMP     DB$DAH          ;Visualización de éste y retorno
02252                   ;
02253                   DP$L:           ;L
02254   0678 3A0904             LDA     DB$Save$L       ;Obtención de valor almacenado
02255   067B C39107             JMP     DB$DAH          ;Visualización de éste y retorno
02256                   ;
02257                   DP$BC:          ;BC
02258   067E 2A4804             LHLD    DB$Save$C       ;Obtención de valor almacenado
02259   0681 C38707             JMP     DB$DHLH         ;Visualización de éste y retorno
02260                   ;
02261                   DP$DE:          ;DE
02262   0684 2A4604             LHLD    DB$Save$E       ;Obtención de valor almacenado
02263   0687 C38707             JMP     DB$DHLH         ;Visualización de éste y retorno
02264                   ;
02265                   DP$HL:          ;HL
02266   068A 2A0904             LHLD    DB$Save$HL      ;Obtención de valor almacenado
02267   068D C38707             JMP     DB$DHLH         ;Visualización de éste y retorno
02268                   ;
02269                   DP$SP:          ;Apuntador de pila
02270   0690 2A0B04             LHLD    DB$Save$SP      ;Obtención de valor almacenado
02271   0693 C38707             JMP     DB$DHLH         ;Visualización de éste y retorno
02272                   ;
02273                   DP$M:           ;Memoria
02274   0696 2A1304             LHLD    DB$End$Address  ;Incremento de la dirección de finalización para
02275   0699 23                 INX     H               ;  hacer la aritmética más simple
02276   069A 221304             SHLD    DB$End$Address
02277
02278   069D 2A1104             LHLD    DB$Start$Address;Comparación de HL con End$Address
02279   06A0 CD3A07             CALL    DB$M$Check$End  ;Final > comienzo
02280   06A3 DAD106             JC      DB$M$Address$OK ;Error, comienzo > final
02281   06A6 CDD607             CALL    DB$MSGI
02282   06A9 0D0A2A2A20         DB      0DH,0AH,'** ERROR - Start Address > End **',0
02283   06CD C9                 RET
02284                   ;
02285                   DB$M$Next$Line:
02286   06CE CDC107             CALL    DB$CRLF                 ;Lanzar a salida ret. carro, salto de línea
02287                   DB$M$Address$OK:                        ;Ignorar CR,LF en la primera línea
02288   06D1 CDD607             CALL    DB$MSGI                 ;Línea indentada
02289   06D4 202000             DB      '  ',0
02290   06D7 2A1104             LHLD    DB$Start$Address        ;Obtención comienzo dirección de línea
02291   06DA CD8707             CALL    DB$DHLH                 ;Visualización en hexadecimal
02292
02293   06DD CDC807             CALL    DB$Colon                ;Visualización de ' : '
02294
02295   06E0 2A1104             LHLD    DB$Start$Address
02296                   DB$M$Next$Hex$Byte:                     ;Almacenamiento dirección de memoria
02297   06E3 E5                 PUSH    H
02298   06E4 CDD007             CALL    DB$Blank                ;Lanzar a salida un blanco
02299   06E7 E1                 POP     H                       ;Recuperación dirección de byte en curso
02300   06E8 7E                 MOV     A,M                     ;Obtención de byte de la memoria
02301   06E9 23                 INX     H                       ;Actualización apuntador a memoria
02302   06EA E5                 PUSH    H                       ;Salvar para utilización posterior
02303   06EB CD9107             CALL    DB$DAH                  ;Visualización en hexadecimal
02304   06EE E1                 POP     H                       ;Recuperación de direc. actualizada de mem.
```

Figura 10-2. (Continuación.)

```
02305   06FF CD3A07             CALL    DB$M$Check$End          ;Comparación HL con dirección de finalización
02306   06F2 CAFE06             JZ      DB$M$Display$ASCII      ;Sí, final de área
02307   06F5 7D                 MOV     A,L                     ;Comprobación de comienzo de nueva línea
02308   06F6 E60F               ANI     0000$1111B              ;  (¿la dirección es XXX0H?)
02309   06F8 CAFE06             JZ      DB$M$Display$ASCII      ;Sí
02310   06FB C3E306             JMP     DB$M$Next$Hex$Byte      ;No, vuelta atrás a por otro
02311                   ;
02312                   DB$M$Display$ASCII:                     ;Visualización de byte en ASCII
02313   06FE CDC807             CALL    DB$Colon                ;Visualización de ' : '
02314   0701 2A1104             LHLD    DB$Start$Address        ;Comienzo ASCII como al principio de línea
02315                   DB$M$Next$ASCII$Byte:
02316   0704 7E                 MOV     A,M                     ;Obtención de byte de la memoria
02317   0705 E5                 PUSH    H                       ;Almacenar dirección de memoria
02318   0706 E67F               ANI     0111$1111B              ;Eliminar paridad
02319   0708 4F                 MOV     C,A                     ;Preparación para salida
02320   0709 FE20               CPI     ' '                     ;Comprobación de no-gráfico
02321   070B D21007             JNC     DB$M$Display$Char       ;Car >= espacio
02322   070E 0E2E               MVI     C,'.'                   ;Visualización no gráfica como '.'
02323                   DB$M$Display$Char:
02324   0710 FE7F               CPI     7FH                     ;Comprobación de DEL (puede ser no-gráfico)
02325   0712 C21707             JNZ     DB$M$Not$DEL            ;No, es gráfico
02326   0715 0E2E               MVI     C,'.'                   ;Forzar '.'
02327                   ;
02328                   DB$M$Not$DEL:
02329   0717 CD0A05             CALL    DB$CONOUT               ;Visualización de carácter
02330   071A E1                 POP     H                       ;Recuperación dirección de memoria
02331   071B 23                 INX     H                       ;Actualización apuntador de memoria
02332   071C 221104             SHLD    DB$Start$Address        ;Actualización copia de memoria
02333   071F CD3A07             CALL    DB$M$Check$End          ;Comprobación de final de volcado de memoria
02334   0722 CA3707             JZ      DB$M$Exit               ;Sí, finalizado
02335   0725 7D                 MOV     A,L                     ;Comprobación de final de línea
02336   0726 E60F               ANI     0000$1111B              ;  comprobando dirección XXX0H
02337   0728 CACE06             JZ      DB$M$Next$Line          ;Sí, comienzo de nueva línea
02338   072B 7D                 MOV     A,L                     ;Comprobación de necesidad de blancos extra
02339   072C E603               ANI     0000$0011B              ;  si la dirección es múltiplo de cuatro
02340   072E C20407             JNZ     DB$M$Next$ASCII$Byte    ;No, vuelta atrás a por nuevo carácter
02341   0731 CDD007             CALL    DB$Blank                ;Sí, salida de blanco
02342   0734 C30407             JMP     DB$M$Next$ASCII$Byte    ;Vuelta atrás a por nuevo carácter
02343
02344                   ;
02345                   DB$M$Exit:
02346   0737 C3C107             JMP     DB$CRLF                 ;Salida de retorno de carro,
02347                                                           ;  salto de línea y retorno
02348                   ;
02349                   DB$M$Check$End:                         ;Comparación de HL con End$Address
02350   073A D5                 PUSH    D                       ;Almacenar DE (salvaguarda)
02351   073B EB                 XCHG                            ;DE = dirección en curso
02352   073C 2A1304             LHLD    DB$End$Address          ;Obtención dirección de finalización
02353   073F 7A                 MOV     A,D                     ;Comparación de bytes MS
02354   0740 BC                 CMP     H
02355   0741 C24607             JNZ     DB$M$Check$End$X        ;Salida si son distintos
02356   0744 7B                 MOV     A,E                     ;Comparación de los bytes LS
02357   0745 BD                 CMP     L
02358                   DB$M$Check$End$X:
02359   0746 EB                 XCHG                            ;HL = dirección en curso
02360   0747 D1                 POP     D                       ;Recuperación de DE
02361   0748 C9                 RET                             ;Retorno con indicadores de condición a uno
02362                   ;
02363                   DP$B$BC:        ;(BC)
02364   0749 2A4804             LHLD    DB$Save$C       ;Obtención palabra-valor salvada
02365   074C 7E                 MOV     A,M             ;Obtención del byte direccionado por ésta
02366   074D C39107             JMP     DB$DAH          ;Visualización del mismo y retorno
02367                   ;
02368                   DP$B$DE:        ;(DE)
02369   0750 2A4604             LHLD    DB$Save$E       ;Obtención palabra-valor salvada
02370   0753 7E                 MOV     A,M             ;Obtención del byte direccionado por ésta
02371   0754 C39107             JMP     DB$DAH          ;Visualización del mismo y retorno
02372                   ;
02373                   DP$B$HL:        ;(HL)
02374   0757 2A0904             LHLD    DB$Save$HL      ;Obtención palabra-valor salvada
02375   075A 7E                 MOV     A,M             ;Obtención del byte direccionado por ésta
02376   075B C39107             JMP     DB$DAH          ;Visualización del mismo y retorno
02377                   ;
02378                   DP$W$BC:        ;(BC+1),(BC)
02379   075E 2A4804             LHLD    DB$Save$C       ;Obtención palabra-valor salvada
02380   0761 5E                 MOV     E,M             ;Obtención de palabra direccionada por éste
02381   0762 23                 INX     H
```

Figura 10-2. (Continuación.)

```
02382    0763 56              MOV     D,M
02383    0764 EB              XCHG                    ;HL = palabra a visualizar
02384    0765 C38707          JMP     DB$DHLH         ;Visualización del mismo
02385                 ;
02386                 DP$W$DE:        ;(DE+1),(DE)
02387    0768 2A4604          LHLD    DB$Save$E       ;Obtención palabra-valor salvada
02388    076B 5E              MOV     E,M             ;Obtención del byte direccionado por ésta
02389    076C 23              INX     H
02390    076D 56              MOV     D,M
02391    076E EB              XCHG                    ;HL = palabra a visualizar
02392    076F C38707          JMP     DB$DHLH         ;Visualización del mismo
02393                 ;
02394                 DP$W$HL:        ;(HL+1),(HL)
02395    0772 2A0904          LHLD    DB$Save$HL      ;Obtención palabra-valor salvada
02396    0775 5E              MOV     E,M             ;Obtención del byte direccionado por ésta
02397    0776 23              INX     H
02398    0777 56              MOV     D,M
02399    0778 EB              XCHG                    ;HL = palabra a visualizar
02400    0779 C38707          JMP     DB$DHLH         ;Visualización del mismo
02401                 ;
02500                 ;#
02501                 ;       DB$Display$CALLA
02502                 ;       Esta rutina visualiza en hexadecimal la DB$Call$Address seguida
02503                 ;       de " : ".
02504                 ;
02505                 DB$Display$CALLA:
02506    077C E5              PUSH    H               ;Salvar Hl del llamador
02507    077D 2A0F04          LHLD    DB$Call$Address ;Obtención de la dirección de llamada
02508    0780 CD8707          CALL    DB$DHLH         ;Visualización de Hl en hexadecimal
02509    0783 E1              POP     H               ;Recuperación de HL del llamador
02510    0784 C3C807          JMP     DB$Colon        ;Visualización de " : " y retorno
02511                 ;
02600                 ;#
02601                 ;
02602                 ;       DB$DHLH
02603                 ;       Visualización de HL en hexadecimal
02604                 ;
02605                 ;       Parámetros de entrada
02606                 ;
02607                 ;               HL = valor a visualizar
02608                 ;
02609                 DB$DHLH:
02610    0787 E5              PUSH    H               ;Almacenar valor de entrada
02611    0788 7C              MOV     A,H             ;Obtención primer byte MS
02612    0789 CD9107          CALL    DB$DAH          ;Visualización de A en hexadecimal
02613    078C E1              POP     H               ;Recuperación valor de entrada
02614    078D 7D              MOV     A,L             ;Obtención byte LS
02615    078E C39107          JMP     DB$DAH          ;Visualización y retorno
02616                 ;
02700                 ;#
02701                 ;
02702                 ;       DB$DAH
02703                 ;       Visualización del registro A en hexadecimal
02704                 ;
02705                 ;       Parámetros de entrada
02706                 ;
02707                 ;               A = valor a convertir y lanzar a la salida
02708                 ;
02709                 DB$DAH:
02710    0791 F5              PUSH    PSW             ;Realizar copia del valor a convertir
02711    0792 0F              RRC                     ;Desplazamiento de A cuatro lugares a la derecha
02712    0793 0F              RRC
02713    0794 0F              RRC
02714    0795 0F              RRC
02715    0796 CDB407          CALL    DB$Nibble$To$Hex        ;Conversión de los 4 bits LS a ASCII
02716    0799 CD0A05          CALL    DB$CONOUT               ;Visualización del carácter
02717    079C F1              POP     PSW             ;Obtención del valor original
02718    079D CDB407          CALL    DB$Nibble$To$Hex        ;Conversión a ASCII de los 4 bits LS
02719    07A0 C30A05          JMP     DB$CONOUT               ;Visualización y retorno al llamador
02800                 ;#
02801                 ;
02802                 ;       DB$CAH
02803                 ;       Conversión del registro A en ASCII hexadecimal y almacenamiento
02804                 ;       en dirección especificada
02805                 ;
02806                 ;       Parámetros de entrada
02807                 ;
```

Figura 10-2. (Continuación.)

```
02808                   ;               A = valor a convertir y lanzar a salida
02809                   ;               HL -> área de mem. int. para recibir dos caracteres de salida
02810                   ;
02811                   ;       Parámetros de salida
02812                   ;
02813                   ;               HL -> byte que sigue al último byte hexadecimal salido
02814                   ;
02815                   DB$CAH:
02816   07A3 F5                 PUSH    PSW             ;Realización de copia del valor a convertir
02817   07A4 0F                 RRC                     ;Desplazamiento de A 4 lugares a la derecha
02818   07A5 0F                 RRC
02819   07A6 0F                 RRC
02820   07A7 0F                 RRC
02821   07A8 CDB407             CALL    DB$Nibble$To$Hex        ;Conversión a hexadecimal ASCII
02822   07AB 77                 MOV     M,A             ;Almacenar en memoria
02823   07AC 23                 INX     H               ;Actualización de apuntador
02824   07AD F1                 POP     PSW             ;Obtención de valor hexadecimal
02825   07AE CDB407             CALL    DB$Nibble$To$Hex        ;Conversión a hexadecimal ASCII
02826   07B1 77                 MOV     M,A             ;Almacenar en memoria
02827   07B2 23                 INX     H               ;Actualización de apuntador
02828   07B3 C9                 RET
02900                   ;#
02901                   ;
02902                   ;       Subrutinas inferiores
02903                   ;
02904                   ;
02905                   ;       DB$Nibble$To$Hex
02906                   ;       Es una subrutina menor que convierte los 4 bits
02907                   ;       menos significativos del registro A en un carácter hexadecimal
02908                   ;       en A y C
02909                   ;
02910                   ;       Parámetros de entrada
02911                   ;
02912                   ;               A = cuarteto a convertir en los 4 bits LS
02913                   ;
02914                   ;       Parámetros de salida
02915                   ;
02916                   ;               A,C = carácter ASCII hexadecimal
02917                   ;
02918                   DB$Nibble$To$Hex:
02919   07B4 E60F               ANI     0000$1111B      ;Aislamiento de los cuatro bits LS
02920   07B6 C630               ADI     '0'             ;Conversión a ASCII
02921   07B8 FE3A               CPI     '9' + 1         ;Comparación con el máximo
02922   07BA DABF07             JC      DB$NTH$Numeric  ;No es necesaria conversión en A -> F
02923   07BD C607               ADI     7               ;Conversión en letra
02924                   DB$NTH$Numeric:
02925   07BF 4F                 MOV     C,A             ;Por conveniencias de otras rutinas
02926   07C0 C9                 RET
02927
02928
02929                   ;
02930                   ;       DB$CRLF
02931                   ;       Rutina simple para visualización de retorno de carro y salto de línea
02932                   ;
02933                   DB$CRLF:
02934   07C1 CDD607             CALL    DB$MSGI         ;Visualización de mensaje
02935   07C4 0D0A00             DB      0DH,0AH,0
02936   07C7 C9                 RET
02937                   ;
02938                   ;       DB$Colon
02939                   ;       Rutina simple para visualización de ' :
02940                   ;
02941                   DB$Colon:
02942   07C8 CDD607             CALL    DB$MSGI         ;Visualización de mensaje
02943   07CB 203A2000           DB      ' : ',0
02944   07CF C9                 RET
02945                   ;
02946                   ;       DB$Blank
02947                   ;       Rutina simple para visualización de ' '.
02948                   ;
02949                   DB$Blank:
02950   07D0 CDD607             CALL    DB$MSGI         ;Visualización de mensaje
02951   07D3 2000               DB      ' ',0
02952   07D5 C9                 RET
03100                   ;#
03101                   ;
03102                   ;       Subrutinas de procesamiento de mensajes
```

Figura 10-2. (Continuación.)

```
03103                   ;
03104                   ;       DB$MSGI  (mensaje en-línea)
03105                   ;       Lanza a salida el mensaje terminado por un byte nulo que sigue
03106                   ;       a la llamada a MSGOUTI
03107                   ;
03108                   ;       Secuencia de llamada
03109                   ;
03110                   ;               CALL    DB$MSGI
03111                   ;               DB      'Message',0
03112                   ;               ... instrucción siguiente
03113                   ;
03114                   ;       Parámetros de salida
03115                   ;               HL -> instrucción siguiente al mensaje
03116                   ;
03117                   ;
03118                   DB$MSGI:
03119                                                   ;Obtención de dirección de retorno de la pila,
03120                                                   ;  almacenar HL del mismo en cabecera de la pila
03121   07D6 E3                 XTHL                    ;HL -> mensaje
03122
03123   07D7 F5                 PUSH    PSW             ;Salvar todos los registros de usuario
03124   07D8 C5                 PUSH    B
03125   07D9 D5                 PUSH    D
03126                   DB$MSGI$Next:
03127   07DA 7E                 MOV     A,M             ;Obtención siguiente byte de datos
03128   07DB 23                 INX     H               ;Actualización apuntador a mensaje
03129   07DC B7                 ORA     A               ;Comprobación de byte nulo
03130   07DD C2E507             JNZ     DB$MSGIC        ;No, continúe
03131
03132   07E0 D1                 POP     D               ;Recuperación registros de usuario
03133   07E1 C1                 POP     B
03134   07E2 F1                 POP     PSW
03135   07E3 E3                 XTHL                    ;Recuperación HL del usuario de la pila,
03136                                                   ;  reemplazándolo por dirección de retorno actualizada
03137   07E4 C9                 RET                     ;Retorno a la dirección que sigue al byte nulo
03138                                                   ;  después del mensaje en-línea
03139                   DB$MSGIC:
03140   07E5 E5                 PUSH    H               ;Almacenar apuntador a mensaje
03141   07E6 4F                 MOV     C,A             ;Preparado para salida
03142   07E7 CD0A05             CALL    DB$CONOUT
03143   07EA E1                 POP     H               ;Recuperación apuntador a mensaje
03144   07EB C3DA07             JMP     DB$MSGI$Next    ;Vuelta atrás a por carácter siguiente
03145
03146                   ;
03147                   ;       DB$MSG
03148                   ;       Salida de mensaje terminado por byte nulo
03149                   ;
03150                   ;       Secuencia de llamada
03151                   ;
03152                   ;       MENSAJE:        DB      'Message',0
03153                   ;               :
03154                   ;               LXI     H,MESSAGE
03155                   ;               CALL    DB$MSG
03156                   ;
03157                   ;       Parámetros de salida
03158                   ;               HL -> byte terminal nulo
03159                   ;
03160                   ;
03161                   DB$MSG:
03162   07EE F5                 PUSH    PSW             ;Salvar registros de usuario
03163   07EF C5                 PUSH    B
03164   07F0 D5                 PUSH    D
03165                   DB$MSG$Next:
03166   07F1 7E                 MOV     A,M             ;Obtención byte siguiente para salida
03167   07F2 B7                 ORA     A               ;Comprobación de terminación byte-00
03168   07F3 CA0008             JZ      DB$MSG$X        ;Salida
03169   07F6 23                 INX     H               ;Actualización apuntador a mensaje
03170   07F7 E5                 PUSH    H               ;Almacenamiento apuntador actualizado
03171   07F8 4F                 MOV     C,A             ;Preparado para salida
03172   07F9 CD0A05             CALL    DB$CONOUT
03173   07FC E1                 POP     H               ;Recuperación apuntador a mensaje
03174   07FD C3F107             JMP     DB$MSG$Next     ;Vuelta atrás a por carácrer siguiente
03175                   ;
03176                   DB$MSG$X:
03177   0800 D1                 POP     D               ;Recuperación registros de usuario
03178   0801 C1                 POP     B
03179   0802 F1                 POP     PSW
```

Figura 10-2. (Continuación.)

```
03180    0803 C9                    RET
03300                    ;#
03301                    ;
03302                    ;       Rutina de depuración de entrada
03303                    ;
03304                    ;       Esta rutina ayuda a la depuración de programas en los que existen
03305                    ;       normalmente instrucciones de entrada. El código de operación de la
03306                    ;       instrucción IN debe reemplazarse por el valor OE7H (RST 4).
03307                    ;
03308                    ;       Esta rutina toma el número de puerto contenido
03309                    ;       en el byte que sigue a la RST 4, lo convierte en
03310                    ;       hexadecimal y lanza el mensaje:
03311                    ;
03312                    ;               Entrada desde puerto XX :
03313                    ;
03314                    ;       Acepta entonces dos caracteres (en hex.) del teclado,
03315                    ;       los convierte a binario en A y devuelve el control al byte
03316                    ;       que sigue al número de puerto
03317                    ;
03318                    ;       *******
03319                    ;       ATENCION - Esta rutina efectúa llamadas a BDOS y DB$CONOUT
03320                    ;       *******
03321                    ;
03322    0804 496E707574DBIN$Message:    DB      'Input from Port '
03323    0814 5858203A20DBIN$Port:       DB      'XX : ',0
03324                    ;
03325                    ;
03326                    DB$Input:
03327    081A 220904             SHLD    DB$Save$HL          ;Salvar HL de usuario
03328    081D E1                 POP     H                   ;Recuperación dirección de número de puerto
03329    081E 2B                 DCX     H                   ;Copia para apuntar a RST
03330    081F 220F04             SHLD    DB$Call$Address     ;Almacenamiento para visualización posterior
03331    0822 23                 INX     H                   ;Vuelta a apuntar a número de puerto
03332                                                        ;Nota: A no necesita preservarse
03333    0823 7E                 MOV     A,M                 ;Obtención de número de puerto
03334    0824 23                 INX     H                   ;Act. direc. de retorno para saltar núm. de puerto
03335    0825 220D04             SHLD    DB$Save$RA          ;Almacenar dirección de retorno
03336    0828 C5                 PUSH    B                   ;Almacenar resto de registros
03337    0829 D5                 PUSH    D
03338    082A F5                 PUSH    PSW                 ;Almacenar número de puerto para uso posterior
03339
03340
03341    082B CDB108             CALL    DB$Flag$Save$On     ;Almacenamiento de estado en curso del indicador
03342                                                        ;  de depuración y activación salida de depuración
03343
03344    082E CDC107             CALL    DB$CRLF             ;Visualización de retorno de carro y salto de línea
03345    0831 CD7C07             CALL    DB$Display$CALLA    ;Visualización dirección de llamada
03346    0834 F1                 POP     PSW                 ;Recuperación número de puerto
03347    0835 211408             LXI     H,DBIN$Port
03348    0838 CDA307             CALL    DB$CAH              ;Conversión a hex. y almacenamiento en mensaje
03349    083B 210408             LXI     H,DBIN$Message      ;Salida de mensaje de indicación
03350    083E CDEE07             CALL    DB$MSG
03351    0841 0E02               MVI     C,2                 ;Obtención de valor de 2 dígitos hexadecimales
03352    0843 CDCF08             CALL    DB$GHV              ;Retorno de valor en HL
03353    0846 7D                 MOV     A,L                 ;Obtención de un byte únicamente
03354
03355    0847 CDBF08             CALL    DB$Flag$Restore     ;Vuelta a situar la salida de depuración a su
03356                                                        ;  estado anterior
03357    084A D1                 POP     D                   ;Recuperación de registros
03358    084B C1                 POP     B
03359    084C 2A0904             LHLD    DB$Save$HL          ;Obtención de HL anterior
03360    084F E5                 PUSH    H                   ;Puesta en cabecera de pila
03361    0850 2A0D04             LHLD    DB$Save$RA          ;Obtención dirección de retorno
03362    0853 E3                 XTHL                        ;TOS = dirección de retorno, HL = valor anterior
03363    0854 C9                 RET
03500                    ;#
03501                    ;
03502                    ;       Rutina de depuración de salida
03503                    ;
03504                    ;       Esta rutina ayuda a la depuración de programas en los que existen
03505                    ;       normalmente instrucciones de salida. El código de operación
03506                    ;       de la instrucción OUT debe reemplazarse por el valor OEFH (RST 5).
03507                    ;
03508                    ;       Esta rutina toma el valor del número de puerto
03509                    ;       contenido en el byte que sigue a la RST 5, lo convierte
03510                    ;       en hexadecimal y lanza el mensaje:
03511                    ;
```

Figura 10-2. (Continuación.)

```
03512                   ;               Salida al puerto XX : AA
03513                   ;
03514                   ;       donde AA es el contenido del registro A antes de que
03515                   ;       se ejecutase la RST 5.
03516                   ;       Se devuelve el control al byte que sigue al número de puerto
03517                   ;
03518                   ;       *******
03519                   ;       ATENCION - Esta rutina efectúa llamadas a BDOS y DB$CONOUT
03520                   ;       *******
03521                   ;
03522                   ;
03523   0855 4F75747075DBO$Message:     DB      'Output to Port '
03524   0864 5858203A20DBO$Port:        DB      'XX : '
03525   0869 414100     DBO$Value:      DB      'AA',0
03526                   ;
03527                   ;
03528                   DB$Output:
03529   086C 220904             SHLD    DB$Save$HL          ;Salvar HL de usuario
03530   086F E1                 POP     H                   ;Recuperación dirección de número de puerto
03531   0870 2B                 DCX     H                   ;Copia para apuntar a RST
03532   0871 220F04             SHLD    DB$Call$Address     ;Almacenamiento para visualización posterior
03533   0874 23                 INX     H                   ;Vuelta a apuntar a número de puerto
03534   0875 324B04             STA     DB$Save$A           ;Preservar valor a lanzar a salida
03535   0878 7E                 MOV     A,M                 ;Obtención de número de puerto
03536   0879 23                 INX     H                   ;Act. direc. de retorno para saltar núm. de puerto
03537   087A 220D04             SHLD    DB$Save$RA          ;Almacenar dirección de retorno
03538   087D C5                 PUSH    B                   ;Almacenar resto de registros
03539   087E D5                 PUSH    D
03540   087F F5                 PUSH    PSW                 ;Almacenar número de puerto para uso posterior
03541
03542   0880 CDB108             CALL    DB$Flag$Save$On ;Almacenamiento de estado en curso del indicador de
03543                                                   ;  depuración y activación salida de depuración
03544
03545   0883 CDC107             CALL    DB$CRLF             ;Visualización de retorno de carro y salto de línea
03546   0886 CD7C07             CALL    DB$Display$CALLA;Visualización dirección de llamada
03547   0889 F1                 POP     PSW                 ;Recuperación número de puerto
03548   088A 216408             LXI     H,DBO$Port
03549   088D CDA307             CALL    DB$CAH              ;Conversión a hex. y almacenamiento en mensaje
03550
03551   0890 3A4B04             LDA     DB$Save$A
03552   0893 216908             LXI     H,DBO$Value         ;Conversión del valor de salida
03553   0896 CDA307             CALL    DB$CAH              ;Conversión a hex. y almacenar en mensaje
03554
03555   0899 215508             LXI     H,DBO$Message       ;Salida de mensaje de indicación
03556   089C CDEE07             CALL    DB$MSG
03557
03558   089F CDBF08             CALL    DB$Flag$Restore ;Vuelta a situar la salida de depuración a su
03559                                                   ;  estado anterior
03560   08A2 D1                 POP     D                   ;Recuperación de registros
03561   08A3 C1                 POP     B
03562   08A4 2A0904             LHLD    DB$Save$HL          ;Obtención de HL anterior
03563   08A7 E5                 PUSH    H                   ;Puesta en cabecera de pila
03564   08A8 2A0D04             LHLD    DB$Save$RA          ;Obtención dirección de retorno
03565   08AB E3                 XTHL                        ;TOS = dirección de retorno, HL = valor anterior
03566   08AC 3A4B04             LDA     DB$Save$A           ;Recuperación de A (NOTA: Indicador no recuperado)
03567   08AF C9                 RET
03700                   ;*
03701                   ;
03702                   ;       DB$Flag$Save$On
03703                   ;       Esta rutina es utilizada únicamente por DB$IN/OUT.
03704                   ;       Almacenar el estado en curso del indicador de control de depuración
03705                   ;       D$Flag, y lo activa para asegurar que la salida DB$IN/OUT
03706                   ;       se lanza en todos los casos.
03707                   ;
03708   08B0 00         DB$Flag$Previous:       DB      0       ;Valor previo del indicador
03709                   ;
03710                   DB$Flag$Save$On:
03711   08B1 F5                 PUSH    PSW                     ;Almacenamiento de los reg. del llamador
03712   08B2 3A0604             LDA     DB$Flag                 ;Obtención del valor en curso
03713   08B5 32B008             STA     DB$Flag$Previous        ;Almacenamiento del mismo
03714   08B8 3EFF               MVI     A,0FFH                  ;Activación indicador
03715   08BA 320604             STA     DB$Flag
03716   08BD F1                 POP     PSW
03717   08BE C9                 RET
03800                   ;*
03801                   ;
```

Figura 10-2. (Continuación.)

```
03802                   ;       DB$Flag$Restore
03803                   ;       Esta subrutina es utilizada únicamente por DB$IN/OUT.
03804                   ;       Vuelve a su estado inicial al indicador
03805                   ;       DB$Flag.
03806                   ;
03807                   DB$Flag$Restore:
03808   08BF F5                 PUSH    PSW
03809   08C0 3AB008             LDA     DB$Flag$Previous        ;Obtención configuración inicial
03810   08C3 320604             STA     DB$Flag                 ;Fijación indicador de control de
03811   08C6 F1                 POP     PSW                     ;  depuración
03812   08C7 C9                 RET
03813
03814                   ;
03900                   ;#
03901                   ;
03902                   ;       Obtención de valor hexadecimal
03903                   ;
03904                   ;       Esta rutina lanza a la salida un mensaje de indicación
03905                   ;       y lee entonces en el teclado un valor hexadecimal.
03906                   ;       Es un tanto simplista en la medida en que termina la entrada
03907                   ;       el primer valor no hexadecimal. El número máximo de dígitos a convertir
03908                   ;       se especifica como parámetro de entrada. Si se entra un número mayor
03909                   ;       que el máximo sólo se tomarán en cuenta los últimos cuatro.
03910                   ;
03911                   ;***************************************************************
03912                   ;                       A T E N C I O N
03913                   ;       DB$GHV utilizará al BDOS para realizar la función de lectura
03914                   ;       de consola (#10). Cuidado con el uso de esta rutina
03915                   ;       desde el interior de un BIOS en ejecución.
03916                   ;***************************************************************
03917                   ;
03918                   ;       Parámetros de entrada
03919                   ;
03920                   ;               HL -> mensaje de salida terminado por byte-00
03921                   ;               C = número de dígitos hexadecimales a entrar
03922                   ;
03923                   ;
03924                   DB$GHV$Buffer:                  ;Memoria intermedia de entrada para caracteres de
03925                   DB$GHV$Max$Count:               ;  consola
03926   08C8 00                 DB      0               ;Fijación del número máximo de caracteres
03927                                                   ;  a entrar
03928                   DB$GHV$Input$Count:
03929   08C9 00                 DB      0               ;Fijación, por el BDOS, de número real de
03930                                                   ;  caracteres entrados
03931                   DB$GHV$Data$Bytes
03932   08CA                    DS      5               ;Zona de memoria intermedia para caracteres
03933                   ;
03934                   ;
03935                   DB$GHV:
03936   08CF 79                 MOV     A,C                     ;Obtención núm. máximo de caracteres a entrar
03937   08D0 FE05               CPI     5                       ;Comprobación con cuenta máximo
03938   08D2 DAD708             JC      DB$GHV$Count$OK         ;Acarreo a uno si A < 5
03939   08D5 3E04               MVI     A,4                     ;Forzar a 4 caracteres únicamente
03940                   DB$GHV$Count$OK:
03941   08D7 32C808             STA     DB$GHV$Max$Count        ;Activación cuenta máx. en mem. int. de ent.
03942   08DA CDEE07             CALL    DB$MSG                  ;Salida de mensaje de indicación
03943   08DD 11C808             LXI     D,DB$GHV$Buffer         ;Aceptación de caracteres de la consola
03944   08E0 0E0A               MVI     C,B$READCONS            ;Código de función
03945   08E2 CD0500             CALL    BDOS
03946
03947   08E5 0E02               MVI     C,B$CONOUT              ;Lanzar salto de línea
03948   08E7 1E0A               MVI     E,0AH
03949   08E9 CD0500             CALL    BDOS
03950
03951   08EC 210000             LXI     H,0                     ;Valor inicial
03952   08EF 11CA08             LXI     D,DB$GHV$Data$Bytes     ;DE -> caracteres de datos
03953   08F2 3AC908             LDA     DB$GHV$Input$Count      ;Obtención cuenta de caracteres de entrada
03954   08F5 4F                 MOV     C,A                     ;Mantener cuenta en C
03955                   DB$GHV$Loop:
03956   08F6 0D                 DCR     C                       ;Decrementar contador
03957   08F7 F8                 RM                              ;Retorno cuando oper. real. (HL tiene el valor)
03958   08F8 1A                 LDAX    D                       ;Obtención carácter siguiente de mem. int.
03959   08F9 13                 INX     D                       ;Actualización apuntador a memoria intermedia
03960   08FA CD1B09             CALL    DB$A$To$Upper           ;Conversión de A a mayúsculas si es necesario
03961   08FD FE30               CPI     '0'                     ;Comprobación de si menor que cero
03962   08FF D8                 RC                              ;Sí, final
03963   0900 FE3A               CPI     '9' + 1                 ;Comprobación de si > 9
03964   0902 DA1009             JC      DB$GHV$Hex$Digit        ;No, debe ser numérico
```

Figura 10-2. (Continuación.)

```
03965    0905 FE41              CPI     'A'                ;Comprobación de si < 'A'
03966    0907 D8                RC                         ;Sí, finalización
03967    0908 FE47              CPI     'F' + 1            ;Comprobación de si > 'F'
03968    090A D0                RNC                        ;Sí, finalización
03969    090B D637              SUI     'A' - 10           ;Conversión de A hasta F a numérico
03970    090D C31209            JMP     DB$GHV$Shift$Left$4 ;Combinar con resultado en curso
03971                   ;
03972                   DB$GHV$Hex$Digit:
03973    0910 D630              SUI     '0'                ;Conversión a binario
03974                   DB$GHV$Shift$Left$4:
03975    0912 29                DAD     H                  ;Desplazamiento de HL cuatro lugares a la
03976    0913 29                DAD     H                  ;  izquierda
03977    0914 29                DAD     H
03978    0915 29                DAD     H
03979    0916 85                ADD     L                  ;Añadir valor bin. a los 4 bits menos sig. de A
03980    0917 6F                MOV     L,A                ;Puesta en HL
03981    0918 C3F608            JMP     DB$GHV$Loop        ;Vuelta atrás por carácter siguiente
04100                   ;*
04101                   ;
04102                   ;       A a mayúsculas
04103                   ;       Convierte el contenido del registro A a mayúscula
04104                   ;       si está en minúsculas
04105                   ;
04106                   ;       Parámetros de entrada
04107                   ;
04108                   ;               A = carácter a convertir
04109                   ;
04110                   ;       Parámetros de salida
04111                   ;
04112                   ;               A = carácter convertido
04113                   ;
04114                   DB$A$To$Upper:
04115    091B FE61              CPI     'a'                ;Comprobación con límite inferior
04116    091D D8                RC                         ;No necesidad de conversión
04117    091E FE7B              CPI     'z' + 1            ;Comparación con límite superior
04118    0920 D0                RNC                        ;No necesidad de conversión
04119    0921 E65F              ANI     5FH                ;Conversión a mayúscula
04120    0923 C9                RET
```

Figura 10-2. (Continuación.)

```
B>ddt fig10-2.hex<cr>
DDT VERS 2.0
NEXT  PC
0924 0000
-g100<cr>

0116 : Flags : Flags = C1Z0M1E1I0
0120 : A Register : A = AA
012F : B Register : B = BB
013E : C Register : C = CC
014D : D Register : D = DD
015C : E Register : E = EE
016B : H Register : H = FF
017A : L Register : L = 11
0189 : Memory Dump #1 : Start, End Address  : 0108, 0128
  0108 :  05 3E AA 01 CC BB 11 EE : .>ª. Ì».n
  0110 :  DD 21 11 FF B7 37 CD 52 05 00 46 6C 61 67 73 00 : Ý!.. ·7ÍR ..Flags.
  0120 :  CD 52 05 02 41 20 52 65 67 : ÍR.. A Reg

01A0 : Memory Dump #2 : Start, End Address  : 0100, 011F
  0100 :  31 6B 03 CD EA 04 CD 15 05 3E AA 01 CC BB 11 EE : 1k.Í ê.Í. .>ª. Ì».n
  0110 :  DD 21 11 FF B7 37 CD 52 05 00 46 6C 61 67 73 00 : Ý!.. ·7ÍR ..Flags.

01B7 : Memory Dump #3 : Start, End Address  : 0101, 0100

** ERROR - Start Address > End **
01CE : Memory Dump #4 : Start, End Address  : 0100, 0100
  0100 :  31 : 1
```

Figura 10-3. Salida de consola desde el programa de prueba de depuración.

```
01E5 : BC Register : BC = BBCC
01F5 : DE Register : DE = DDEE
0205 : HL Register : HL = FF11
0215 : SP Register : SP = 0369
022E : Byte at (BC) : (BC) = BC
023F : Byte at (DE) : (DE) = DE
0250 : Byte at (HL) : (HL) = F1
026A : Word at (BC) : (BC+1),(BC) = 0B0C
027B : Word at (DE) : (DE+1),(DE) = 0D0E
028C : Word at (HL) : (HL+1),(HL) = 0F01
Debug output has been re-enabled.
This message should display 5 times
This message should display 5 times
This message should display 5 times
This message should display 5 times
This message should display 5 times
032B : Input from Port 11 : aa

032D : Output to Port 22 : AA
```

Figura 10-3. (Continuación.)

recibir la orden de enviar un fichero que contenga todos los símbolos del programa, junto con sus direcciones correspondientes. Una vez que el programa ha sido cargado por SID, se puede referir a la imagen de memoria del programa no por dirección, sino por el nombre simbólico real en el código fuente. SID soporta también el concepto de "cuenta pasos" cuando se utilizan puntos de interrupción.

ZSID (Z80 depuración simbólica)
Esta es la versión CPU Z80 de SID. El miniensamblador/desensamblador utiliza nemónicos de instrucción Zilog en vez de los utilizados por Intel.

Puesta en marcha del CP/M por primera vez

Es mucho más difícil introducir un CP/M en un nuevo sistema de cálculo que depurar una versión mejorada en un sistema que funcione ya con CP/M. A menudo se enfrenta uno con un "muro infranqueable" sin ninguna herramienta de depuración adecuada para ayudarle.

Por ejemplo, se instala el sistema CP/M en un disquete (utilizando otra computadora basada en CP/M), se pone el disquete en la nueva computadora y se pulsa el botón RESET. La cabeza del disco carga en el disco y luego ¡nada! No se puede utilizar ningún programa tal como DDT o SID porque aún no se ha tenido instalado CP/M y funcionando en la nueva computadora. ¿O sí se puede?

La respuesta es, siempre que sea posible, depurar el programa para la nueva máquina en un sistema CP/M existente. Quizá haya que "falsear" algunos aspectos del nuevo cargador o del BIOS de forma que el hecho de la

comprobación del mismo en la máquina principal no interaccione con el CP/M que funciona ya en ella.

Este esquema permite estar prácticamente seguros de la lógica del programa antes de cargar el disquete en la nueva máquina. Ayudará a eliminar problemas causados por problemas del hardware de la nueva computadora.

La situación más difícil de todas se presenta si se tiene sólo la nueva computadora y los discos de Digital Research. La única opción es encontrar una forma de leer en la memoria la imagen del CP/M contenida en el disquete, hacer los arreglos a mano en la nueva consola y en los controladores de disco (lo cual no es tarea fácil), escribir la imagen modificada en un disquete y recurrir a la prueba de Orville Wright.

Si se valora el tiempo, es siempre más efectivo por lo que a coste se refiere utilizar otro sistema con el CP/M ya instalado. Esto es cierto aun cuando los dos sistemas no tengan el mismo formato de disquetes. Se puede ya realizar el cargador y construir la imagen del CP/M en la máquina. Entonces se carga la imagen directamente en la memoria de la nueva máquina y se escribe en un disquete.

Este proceso de *carga inferior* requiere, sin embargo, que la nueva computadora tenga un programa monitor de memoria de sólo lectura (ROM). Dependiendo de la capacidad de este programa monitor en ROM, puede tenerse que hacer arreglos en la memoria de la nueva máquina para que un programa primitivo de "carga inferior" lea caracteres de 8 bits de un puerto serie, amontonándolos en la memoria y devolviendo el control al programa monitor cuando se pulsa un carácter del teclado de la consola de la nueva máquina. De hecho, algunos programas monitor en ROM poseen un programa de carga inferior incorporado.

Depuración del cargador del CP/M

El cargador del CP/M, como se recordará, está escrito en una de las pistas más externas de un disquete o disco rígido. En un disquete estándar de 8 pulgadas, una sola cara y simple densidad, el cargador de CP/M está almacenado en el primer sector de la primera pista. El cargador se lleva a la memoria por medio de firmware, que obtiene el control de la CPU cuando se conecta la máquina o se pulsa el botón de RESET.

El cargador ha de ser compacto, porque el espacio del disquete en el cual está almacenado es limitado: no puede ser mayor de 128 bytes para disquetes estándar de 8 pulgadas. Esto tiende a excluir el uso de las subrutinas de depuración descritas, de forma que habrá que recurrir a técnicas más primitivas.

Comprobación del cargador bajo el CP/M

Un cargador se desarrolla mejor en un sistema basado en CP/M. La tarea es más sencilla si se tiene ya el CP/M funcionando en la nueva máquina y se está preparando simplemente una versión mejorada del cargador. En este caso se puede comprobar la mayor parte de los programas como si fueran programas de usuario funcionando en el área de programas transitorios (TPA).

La mayor parte de cargadores se cargan en la memoria en la posición 0000H, por lo que al principio del programa que se ha de depurar se debe colocar una línea de origen temporal que rece

```
ORG     100H
```

Si se omite esto y se pide al DDT que cargue el fichero HEX sacado por el ensamblador, cargará en el origen real, 0000H, y borrará los contenidos de la página base para la versión del CP/M que se esté utilizando. Esto ocasionará una caída del sistema; habrá que pulsar el botón RESET y volver a cargar el CP/M. Cuando ocurre esto, el DDT no dice directamente que hay algo mal; únicamente lanza el mensaje "?" cuando se le pide que cargue el fichero HEX. Se descubrirá que el fichero se ha "marchado" únicamente cuando se intente hacer alguna otra cosa.

Se necesitará también ajustar las direcciones en las que el cargador intenta cargar la imagen del CP/M. Si no se hace así, se escribirá sobre la versión del CP/M que está funcionando en la actualidad.

Con estos ajustes realizados se puede cargar el cargador bajo DDT y comprobar su ejecución, confirmando que carga la imagen correcta en las direcciones correctas para depurar y transferir el control al vector de saltos del BIOS. Cuando todo parezca estar funcionando correctamente, utilícese la instrucción IF para desconectar el programa de depuración, volver a ensamblar el cargador y escribirlo en un disquete. Entonces póngase el disquete en la unidad A y púlsese RESET.

¿Estaba cargado el "cargador"?

En este punto hay que aclarar si el cargador se está cargando en la memoria cuando se conecta la máquina o cuando se pulsa RESET. La mejor forma de hacerlo, y que además se puede dejar situada permanentemente, es enviar un mensaje de "funcionamiento correcto" en cuanto el cargador obtenga el control. Esto requiere hardware fijado para preparar al chip de USART (Universal Synchronous/Asynchronous Receive/Transmit) para enviar datos, aunque algunos fabricantes escriben este código de inicializa-

ción en el firmware que carga el cargador de arranque. Un mensaje adecuado de funcionamiento correcto sería el siguiente:

```
CP/M Bootstrap Loader : Vn 1.0 11/18/82
```

Si no se ve este mensaje, debe suponerse que el control *no* ha sido transferido al cargador. Esto será útil en el futuro si alguien protesta porque el CP/M no puede ser cargado. Si este mensaje no aparece, probablemente no se tiene el CP/M en el disco.

¿Cargó el cargador de autocarga al CP/M?

Esta cuestión es más difícil de resolver que la de si el propio cargador había sido cargado, especialmente si ha aparecido el mensaje de funcionamiento correcto del cargador o entonces cae el sistema. Un mensaje anterior en el proceso de arranque en frío del BIOS puede confirmar la transferencia correcta del control al BIOS.

Si el problema con el programa de autocarga es grave, puede tener que adaptarse la subrutina de depuración de descarga de memoria, descargando los contenidos de memoria en la consola para ver qué información está colocando en la memoria el cargador. Visualizar 100H bytes empezando desde el principio del vector de saltos de BIOS. Esta tabla tiene un diseño inmediatamente reconocible de valores 0C3H cada tres bytes.

Debería comprobarse también que el cargador esté cargando el número correcto de sectores del disco en la memoria. Si carga demasiado pocos, el CP/M puede conectarse sólo para caer unos momentos más tarde porque intente bien ejecutar un programa o acceder a una constante al final del BIOS. Si el cargador carga demasiados sectores del disco, el exceso puede "enrollarse alrededor" del principio de la memoria y escribir sobre el propio cargador, en la posición 0000H, antes de que haya completado su tarea. En este caso solamente se vería la señal de conexión para el cargador, no para el BIOS.

Depuración del BIOS

Mejor que intentar depurar el BIOS como una única pieza de programa, la depuración se hará tratándolo como una serie de módulos funcionales separados.

No obstante las filosofías "top-down" actuales que tratan en primer lugar con toda la estructura, puede ser más rápido depurar las subrutinas de bajo nivel primero en un controlador de dispositivos. Esto proporciona una base sólida sobre la cual construir.

El BIOS puede dividirse en sus módulos constituyentes de la siguiente forma:

Entrada de carácter:
  Servicio de interrupción.
  Servicio de no interrupción.

Salida de caracteres.

Rutinas de interrupción:
  Reloj de tiempo real.
  Temporizador-centinela.

Controladores de disco:
  De alto nivel (desbloqueadores).
  De bajo nivel (entrada/salida física).

Hay que planear escribir un programa de pruebas para cada uno de estos módulos. Este programa de pruebas sirve para dos propósitos: en primer lugar, proporciona un medio de transferir el control al módulo bajo comprobación en una forma controlada. En segundo lugar, incluye los módulos necesarios o módulos falseados para "engañar" al módulo bajo comprobación al responder como si estuviera funcionando en un BIOS completo bajo CP/M.

Utilizando la base de comprobación se puede comprobar cada parte de la lógica de los módulos excepto la parte que puede ser crítica en cuanto a tiempo. Los problemas causados por el tiempo, tales como interrupciones inutilizadas por mucho tiempo o programas demasiado lentos o demasiado rápidos para un chip controlador periférico particular, tienden a mostrarse sólo cuando se está comprobando el hardware definitivo y cuando se está funcionando con un nuevo BIOS bajo CP/M.

Lo que se debe comprobar en el BIOS

Describir detalladamente cómo depurar cada módulo del BIOS llenaría varios volúmenes. Recuérdese que se está intentando establecer la *ausencia* de errores utilizando una técnica que, por su propia naturaleza, tiende a mostrar únicamente su *presencia*.

Existen dos aproximaciones básicas a la depuración. Una es el método laborioso y de perseveración, controlando cada aspecto del programa para asegurarse de que todos los dispositivos trabajan realmente. El segundo es intentar hacer algo útil con el programa.

Planifíquese para utilizar ambos. Comiéncese con el primer método, comprobando cada dispositivo bajo control de la base de comprobación hasta que se esté seguro de que está trabajando *in vitro*. Cuando todos los

módulos del BIOS se hayan comprobado individualmente, constrúyase un sistema CP/M e inténtese realizar algún trabajo útil con él. Inténtese utilizar el sistema para trabajos prácticos, comprobando *in vitro* lo que puede ser un buen control.

Lista de control de características

Hágase una lista de las características específicas incluidas en los distintos módulos BIOS. Entonces proyéctense secuencias de control específicas que muestren que cada una de ellas funciona correctamente.

El mismo programa de base de comprobación puede comprobar a menudo todas las características de un módulo controlador. Si no es así, créese una nueva base de comprobación para las más exóticas.

Manténganse las rutinas de base de comprobación. La experiencia demuestra que a menudo se necesitan poco después de haberlas borrado. Aun después de haber comprobado el BIOS, las rutinas de base de comprobación volverán fácilmente si se decide mejorar más tarde un controlador particular. El programa del controlador se puede extraer del BIOS, pegarlo a la base de comprobación y comprobar el nuevo programa aislado del BIOS.

Las secciones siguientes muestran ejemplos de bases de comprobación para distintos controladores, junto con ejemplos de listas de control. Estas se utilizaron para comprobar las rutinas del BIOS ejemplarizado que se mostró en capítulos anteriores.

Dispositivos de procesamiento de caracteres

La figura 10-4 muestra el programa para una rutina de base de control ejemplo para controladores de entrada/salida de caracteres del BIOS. Este programa será seguido por los controladores de entrada/salida de caracteres, exactamente como aparecerían en el BIOS, salvo que todas las instrucciones IN y OUT serían reemplazadas de nuevo por las del RST 4 y 5 respectivamente (véase figura 10-2), de forma que se puedan introducir valores de entrada e inspeccionar valores de salida en la consola.

Este ejemplo contiene el programa de inicialización para el paquete de depuración que se muestra en la figura 10-2 y el programa de preparación de un RST 6 utilizado para "falsear" interrupciones de caracteres de entrada.

El bucle principal comprobado consiste en una interrupción falseada de carácter de entrada seguida de llamadas opcionales a CONIN o CONOUT, la devolución de control al DDT o un bucle para falsear otra interrupción de carácter. Unicamente se podrá devolver el control al DDT si se usa DDT

```
                        ;       Base de test para controladores de E/S de caracteres en el BIOS
                        ;
                        ;       El fichero fuente completo consiste en tres componentes:
                        ;
                        ;               1. El programa propiamente dicho que se muestra aquí.
                        ;               2. Los controladores de E/S de caracteres destinados al BIOS.
                        ;               3. El paquete de programas de depuración mostrado en la figura 10-2.
                        ;
FFFF =          TRUE    EQU     OFFFFH
0000 =          FALSE   EQU     NOT TRUE

FFFF =          DEBUG   EQU     TRUE            ;Para ensamblaje condicional de las
                                                ;  instrucciones RST en lugar de las IN
                                                ;  y OUT en los controladores
0030 =          RST6    EQU     30H             ;Utilización de RST 6 para simular interrupción
                                                ;  por carácter de entrada
0100                    ORG     100H
                START:
0100 31D101             LXI     SP,Test$Stack   ;Utilización de una pila local
0103 CDD101             CALL    DB$Init         ;Inicialización del paquete de depuración
0106 3EC3               MVI     A,JMP           ;Activar RST 6 con código de operación JMP
0108 323000             STA     RST6
010B 21D101             LXI     H,Character$Interrupt   ;Fijación de la dirección de JMP de RST 6
010E 223100             SHLD    RST6 + 1
                        ;
                        ;       Realizar repetitivamente entradas a la rutina de interrupción
                        ;       de caracteres para asegurar que los caracteres son leídos
                        ;       y almacenados en la memoria intermedia de entrada
                        ;
                Testbed$Loop:
0111 3EAA               MVI     A,0AAH          ;Fijación de registros a una configuración conocida
0113 01CCBB             LXI     B,0BBCCH
0116 11EEDD             LXI     D,0DDEEH
0119 2111FF             LXI     H,0FF11H
011C F7                 RST     6               ;Simulación de interrupción por carácter de entrada

011D CDD101             CALL    DB$MSGI         ;Visualización de mensaje en-línea
0120 0D0A456E74         DB      0DH,0AH,'Enter I to Input Char., O to Output, D to enter
0152 444454203A         DB      'DDT : ',0

0159 CDD101             CALL    DB$CONINU       ;Obtención de carácter en mayúscula
015C FE49               CPI     'I'             ;CONIN?
015E CA7201             JZ      Go$CONIN
0161 FE44               CPI     'D'             ;DDT?
0163 CA6E01             JZ      Go$DDT
0166 FE4F               CPI     'O'             ;CONOUT?
0168 CA9101             JZ      Go$CONOUT
016B C31101             JMP     Testbed$Loop    ;Vuelta atrás para nueva interrupción
                Go$DDT:
016E FF                 RST     7               ;Entrada al DDT (RST 7 activada por DDT)
016F C31101             JMP     Testbed$Loop
                Go$CONIN:
0172 CDD101             CALL    CONST           ;Obtención del estado de la consola
0175 CA1101             JZ      Testbed$Loop    ;No hay datos esperando
0178 CDD101             CALL    CONIN           ;Obtención de datos de la memoria intermedia

017B CDD101             CALL    DB$Display      ;Visualización del carácter devuelto en el
017E 02                 DB      DB$A            ;  registro A
017F 434F4E494E         DB      'CONIN returned',0

018E C37201             JMP     Go$CONIN        ;Repetir el bucle CONIN hasta que no queden
                                                ;  caracteres esperando
                        ;
                Go$CONOUT:
0191 CDD101             CALL    CONST           ;Obtención del estado de la consola
0194 CA1101             JZ      Testbed$Loop    ;No hay datos a la espera
0197 CDD101             CALL    CONIN
019A 4F                 MOV     C,A             ;Preparado para salida
019B CDD101             CALL    CONOUT          ;Salida a la consola
019E C39101             JMP     Go$CONOUT       ;Repetir mientras hayan datos
                ;
01A1 9999999999         DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
01B1 9999999999         DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
01C1 9999999999         DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
```

Figura 10-4. Base de comprobación para controladores de entrada/salida de caracteres del BIOS.

```
                Test$Stack:
                ;
                ;       Rutinas ficticias de aquellas mostradas en otras figuras
                ;
                ;       Rutinas del BIOS (figura 8-10)
                ;
                CONST:                  ;BIOS estado de consola
                CONIN:                  ;BIOS entrada de consola
                CONOUT:                 ;BIOS salida de consola
                Character$Interrupt:    ;Rutina de servicio de interrupción para caracteres de entrada
                ;
                ;     Rutinas de depuración (figura 10-2)
                ;
                DB$Init:                ;Inicialización de depuración
                DB$MSGI:                ;Visualización de mensaje en-línea
                DB$CONINU:              ;Obtención de carácter del teclado en mayúscula
                DB$Display:             ;Rutina principal de visualización de depuración
0002 =          DB$A    EQU     02      ;Programa de visualización, DB$Display
```

Figura 10-4. (Continuación.)

para cargar la base de comprobación y los programas de control en primer lugar. La ejecución de un RST 7 sin utilizar DDT provocará una caída del sistema, porque el DDT fija la instrucción JMP necesaria en la posición 0038H de la página base.

La interrupción de carácter de entrada falseada transfiere el control directamente a la rutina de servicio de interrupción del BIOS (véase el ejemplo de la figura 8-10, línea 04902, etiqueta Character$Interrupt). Esta lee los puertos de estado de cada uno de los dispositivos de caracteres; se pueden introducir los valores del byte de estado específicos que se deseen. Si se introduce un valor que indique que un carácter de datos está "entrando", el valor de 8 bits avisará que está "entrado". Se puede hacer que aparezca la rutina de servicio de interrupción como caracteres de entrada y amontonar caracteres en la memoria intermedia de entrada. Para fines de depuración, redúzcase el tamaño de la memoria intermedia de entrada a ocho bytes. El hacerla mayor significa que habrá que introducir más caracteres para comprobar la lógica umbral de la memoria. Para comprobar la rutina de servicio de interrupción, se pasará a través del bucle de base de comprobación principal no haciendo otra cosa que falsear las interrupciones de caracteres de entrada e introducir valores de estado y datos. Los caracteres de datos se amontonarán entonces en la memoria de entrada.

Para comprobar el funcionamiento correcto de las rutinas de servicio de interrupción, se puede permanecer en control con el DDT desde la salida. Alternativamente se puede utilizar sólo el DDT para cargar el fichero HEX de base de comprobación haciendo un bucle, introduciendo algunos caracteres y después solicitar que la base de comprobación devuelva el control al DDT. Entonces se puede utilizar éste para inspeccionar los contenidos de la tabla(s) de dispositivo y las memorias intermedias de entrada.

Otra posibilidad es crear rutinas de depuración que compongan los

contenidos de la tabla de dispositivos en una forma significativa, con cada campo titulado de la forma siguiente:

```
DEVICE TABLE 0
     Status Port     81   Data Port       80
     Output Ready    01   Input Ready     02
     DTR high        40
     Reset Int. Prt D8    Reset Int. Val. 20
     :
     :
     Status Byte 1
          Output Suspended
          Output Xon Enabled
     :
     :
     Buffer Base OE8C
     Put Offset      05       Get Offset       01
     Char. Count     04       Control Count    00
     Data Buffer
     41 42 43 44 45 00 00 00
```

Esta rutina de tabla de dispositivos requerirá gran cantidad de esfuerzo de programación y depuración —pero será rentable—. Se puede obtener una instantánea de la tabla de dispositivos sin tener que decodificar vaciados de memoria hexadecimal ni bits individuales. Los valores constantes de las tablas de dispositivos también se visualizan, de forma que si un error de programa contamina la tabla, se sabrá inmediatamente.

La próxima sección presenta ejemplos de las comprobaciones específicas que es preciso realizar, junto con una descripción de la estrategia que se puede utilizar.

Lista de control de la rutina de servicio de interrupción En un BIOS en funcionamiento, el control se transfiere al módulo de servicio interruptor siempre que un carácter entrante produce una interrupción. En el BIOS del ejemplo de la figura 8-10 (línea 4900), el programa examina cada dispositivo por turno para determinar cuál de ellos causa la interrupción.

Cuando se está depurando la rutina de servicio de interrupción utilizando instrucciones de entrada/salida "falseadas", habrá que introducir valores de byte de estado específicos. Consúltense las declaraciones de la tabla de dispositivos de la figura 8-10, línea 1500, para determinar qué valores han de introducirse para hacer que la rutina de servicio piense que está llegando un carácter de entrada o que el terminal de datos preparado (DTR) es alto o bajo.

Póngase en marcha el proceso de depuración utilizando la primera tabla de dispositivo. Repítanse entonces las comprobaciones en las otras tablas de dispositivo.

Lo que sigue a continuación es una lista de comprobación de características y aspectos que han de ser comprobados al depurar la rutina de servicio de interrupción:

¿Están todos los registros almacenados de nuevo correctamente a la salida del servicio de interrupción?

Utilizando el DDT, empiécese a ejecutar desde el principio la base de comprobación. Fíjese un punto de ruptura (con la orden G100,nnnn) para obtener la devolución del control inmediatamente antes del CALL Character$Interrupt. Utilícese la orden X para visualizar todos los registros, y entonces, utilizando la orden G,nnnn, se fija un punto de ruptura en la instrucción que sigue inmediatamente al CALL Character$Interrupt. Los controladores de carácter avisarán de los valores de estado. Introdúzcase 00 (que indica que no está introduciéndose ningún carácter). Visualícense los registros nuevamente —sus valores deberían ser los mismos—. Recuérdese comprobar el valor del indicador de pista y del total del área de pistas que ya se ha usado.

NOTA: No debe sorprender demasiado si se pierde el control de la máquina cuando se intenta comprobar por primera vez. Al principio se cometen errores lógicos fundamentales. Si el sistema cae, hay que ponerlo a cero, volver a cargar el CP/M, y entonces comenzar de nuevo la comprobación. Esta vez, mejor que fijar el segundo punto de ruptura en la instrucción que sigue al CALL Character$Interrupt, hay que aventurarse en el programa de Character$Interrupt, y, yendo a través de él con pocas instrucciones a la vez, ir fijando puntos de ruptura antes de cualquier instrucción que pueda provocar una transferencia de control. Calcular hasta dónde se ha llegado en el controlador antes de que salte fuera o se sitúe en un bucle.

¿La rutina de servicio introduce en la pila un número significativo de bytes después de haberse producido una interrupción?

Cuando se devuelve el control después del CALL Character$Interrupt, utilícese la orden D (volcado) para volcar la memoria del área de pista en la consola. Contrólese hasta dónde se ha descendido en la pila buscando el punto donde las constantes que acostumbraban llenar el área de pila se han llenado con otros datos escritos sobre ellas.

El BIOS del ejemplo de la figura 8-10 salva sólo los contenidos del par de registros HL en la pila de preinterrupción. Conecta entonces a una pila de BIOS particular para salvar los contenidos del resto de los registros y servir la interrupción.

¿Se han añadido correctamente los caracteres de datos a la memoria intermedia de entrada?

"Introdúzcase" un carácter que no sea de control por medio de la rutina Character$Interrupt. Contrólense entonces los contenidos de la tabla de dispositivo adecuada. La cuenta de caracteres y el desplazamiento de puesta deberán ponerse a uno. Compruébense

entonces los contenidos de la propia memoria intermedia de entrada; ¿contiene el carácter que se "introdujo"?

¿Se han añadido correctamente los caracteres de control a la memoria intermedia de entrada?

"Introdúzcase" un carácter de control tal como 01H. No deberán utilizarse ETX, ACK, XON o XOFF (03H, 06H, 11H y 13H, respectivamente); éstos pueden causar efectos secundarios si se presentan errores en la lógica de manejo del protocolo. Compruébese que el carácter esté almacenado en el próximo byte de memoria intermedia de entrada y que los contadores de caracteres y de control están puestos a dos y uno, respectivamente. El desplazamiento de puesta deberá ser puesto también a dos.

Cuando el umbral de llenado de la memoria de entrada se ha alcanzado, ¿envía el controlador el carácter de protocolo correcto?

Colóquese el primer byte de estado en la primera tabla de dispositivo para activar la entrada del protocolo XON o RTS, o ambos. Luego hay que ir alrededor del bucle de la base de comprobación principal colocando caracteres en la memoria intermedia de entrada. Contrólense los valores visualizados en la consola para ver si los controladores envían los valores correctos cuando la memoria intermedia está casi llena (umbral por defecto cuando quedan cinco bytes). El controlador debería bajar entonces la línea RTS o enviar un carácter XOFF o ambas cosas, según el protocolo de entrada que se haya activado.

Cuando la memoria intermedia de entrada está completamente llena, ¿responde el controlador correctamente?

Esto es una extensión de la comprobación anterior. Introdúzcase un carácter más de los que se pueden encajar en la memoria intermedia. Compruébese para ver que los controladores no acumulan el carácter en la memoria intermedia de entrada y que un carácter BELL (07H) se envía al puerto de datos.

¿Los caracteres de protocolo XON/XOFF son reconocidos y los indicadores de control necesarios fijados o puestos a cero?

Vuélvanse a cargar la base de comprobación y los controladores. Fíjese el byte de estado para facilitar la salida del protocolo XON/XOFF. Usese entonces la rutina Character$Interrupt para introducir un carácter XOFF (13H). Compruébese para ver si el carácter XOFF no se ha situado en la memoria de entrada. En lugar de ello, el byte de estado ha de activarse para indicar que la salida ha sido suspendida.

Introducir un XON y comprobar para ver si el indicador de salida suspendida ha sido puesto a cero.

¿Detecta el controlador y pone a cero los errores de hardware correctamente?

Procédase como si se fuera a introducir un carácter en la memoria intermedia de entrada, pero en su lugar introdúzcase un valor de byte de estado que indique que ha ocurrido un error de hardware (introdúzcase el valor dado en la tabla de dispositivo para DT$Detect$Error$Value).

Compruébese que el controlador detecta el estado de error y envía el valor correcto de puesta a cero del error al puerto de control adecuado.

Lista de control de rutina de servicio de no interrupción En un BIOS "viviente", las rutinas de servicio de no interrupción son accesibles por medio de los puntos de entrada de CONIN y de CONST en el vector de saltos del BIOS. Durante la depuración, la base de comprobación puede llamar al programa CONIN y a CONST directamente.

¿Funciona el redireccionamiento de entrada? ¿Llega el control al controlador con la tabla de dispositivo correctamente seleccionada?

Esto es lo que mejor se comprueba directamente con el DDT. Utilícese la orden Gnnnn,bbbb para transferir el control al programa CONIN con un punto de ruptura en la instrucción RET al final de la rutina Select$Device$Table (véase figura 8-10, línea 04400). Compruébese que el par de registros DE esté señalando a la tabla de dispositivo 0. Si no es así, habrá que volver a comenzar la prueba. Utilícese la orden Tn para obtener la traza de DDT a través de la subrutina Select$Device$Table para encontrar el error.

¿Vuelven los caracteres correctamente desde la memoria intermedia?

Utilícese la base de comprobación para "introducir" un carácter o dos. Utilícese entonces la base de comprobación para realizar algunas entradas en CONIN. Compruébense los caracteres devueltos por la memoria intermedia.

¿Están las cantidades de caracteres de datos y de caracteres de control correctamente decrementadas?

Después de que cada carácter haya sido trasladado desde la memoria intermedia por CONIN, utilícese el DDT para examinar la tabla de dispositivos y compruébese que las cantidades de caracteres de datos y de caracteres de control se han decrementado correctamente. Compruébese también que el apuntador de obtención ha trasladado hacia arriba la memoria intermedia de entrada.

Cuando la memoria intermedia alcanza el umbral de "casi vacía", ¿emite el controlador el carácter de protocolo correcto o manipula correctamente la petición para enviar línea (RTS)?

Utilícese el DDT para activar la entrada de protocolos RTS o XON o ambos. Entonces introdúzcanse caracteres en la memoria intermedia de entrada hasta que alcance el umbral de llenado (por defecto, cuando sólo quedan cinco bytes de repuesto). Confírmese que se lleva a cabo el proceso de "memoria intermedia casi llena". Entonces háganse llamadas repetitivas a CONIN para hacer que los datos fluyan fuera de la memoria intermedia. Compruébese que el proceso de "vaciado de memoria" ocurre cuando se alcanza el umbral correcto. Para el protocolo RTS, el controlador deberá enviar un valor elevado de RTS al puerto de control RTS especificado. Para XON, el controlador deberá enviar un carácter XON al puerto de datos (después de haber leído el puerto de estado para asegurarse de que el hardware puede enviar el carácter).

¿Maneja el controlador el "enrollamiento" de la memoria intermedia correctamente?

Introdúzcanse caracteres en la memoria intermedia de entrada hasta que esté completamente llena. Realícese entonces una sola llamada CONIN para desplazar el primer carácter de la memoria intermedia. A continuación introdúzcase un carácter más en ella. Compruébese que el apuntador de obtención está puesto a uno y el de colocación puesto a cero.

A continuación háganse llamadas sucesivas CONIN para vaciar la memoria intermedia. Introdúzcase entonces un carácter más y compruébese que este último carácter está situado en el primer byte de la memoria intermedia de entrada.

¿Puede manejar el controlador la "entrada forzada" correctamente?

Utilizando el DDT, fíjese el apuntador de entrada forzada para que indique a una cadena terminada en byte 00; por ejemplo, utilícese una de las cadenas de decodificación de tecla de función por defecto. (En la figura 8-10, el apuntador de entrada forzada está inicializado para apuntar a una "cadena de arranque" —declarada al principio del bloque de configuración en la línea 00400.)

Utilizando el DDT, llámese la rutina CONST y compruébese que vuelve con A=0FFH (indicando que parece ser que hay datos de entrada esperando).

Háganse llamadas sucesivas a CONIN y confírmese que los bytes de datos de la cadena de entrada forzada han vuelto. Compruébese que el forzamiento de entrada termina cuando se detecta el byte 00H.

¿Opera correctamente la rutina de estado de consola cuando comprueba caracteres de datos en la memoria intermedia, caracteres de control en la misma y entrada forzada?

Introdúzcase un carácter único que no sea de control, tal como 41H, dentro de la memoria intermedia de entrada. Utilizando el DDT, compruébese que el segundo byte de estado en la tabla de dispositivo tiene el indicador falseado de mecanografiado en avance puesto a cero. Llámese a la rutina CONST —deberá volver con A=0FFH (lo que quiere decir que hay datos en la memoria intermedia)—. Entonces póngase a uno el bit de mecanografiado en avance falseado en el segundo byte de estado y llámese de nuevo CONST. Deberá volver con A=00H (significando que ahora "no hay datos" en la memoria). Introdúzcase un solo carácter de control en la memoria intermedia. Ahora CONST deberá volver con A=0FFH porque hay un carácter de control.

¿Reconoce el controlador secuencias de escape introducidas desde las teclas de función del teclado?

Esta es una de las características difícil de controlar cuando no está funcionando la rutina de reloj de tiempo real. El controlador utiliza el temporizador-centinela para esperar hasta que todos los caracteres de la secuencia de escape hayan llegado. Sin embargo, habrá que modificar el programa CONIN de forma que el temporizador-centinela aparezca para cronometrar inmediatamente, en vez de esperar el tic-tac del reloj de tiempo real. Para realizar este cambio, consúltese la figura 8-10, línea 2200; es el principio de la rutina CONIN. Búsquese la etiqueta CONIN$Wait$For$Delay. Después de algunas instrucciones hay un JNZ CONIN$Wait$For$-Delay. Utilizando el DDT, póngase a 00H los tres bytes de este JNZ.

Entonces, utilizando la base de comprobación, introdúzcase la secuencia de escape completa en la memoria intermedia de entrada. Por ejemplo, introdúzcanse los valores hexadecimales 1B, 4F, 51 (ESCAPE, O, P), que corresponden a los caracteres emitidos en un terminal VT-100 cuando se pulsa FUNCTION KEY 1 (PF1).

A continuación utilícese la base de comprobación para realizar llamadas sucesivas a CONIN. Deberá verse el texto asociado con la tecla de función (FUNCTION KEY 1, LINE FEED) que son devueltos por CONIN.

Repítase esta prueba utilizando diferentes secuencias de teclas de función, incluyendo una secuencia que no corresponde a ninguna de las teclas de función prefijadas. Compruébese que la propia secuencia de escape sea devuelta por CONIN sin ser cambiada por otra cadena.

¿Puede el controlador ver la diferencia entre una tecla de función y la misma secuencia de escape generada a golpes de tecla discretos?

Esta comprobación es casi igual que la anterior. Hágase el mismo añadido al programa CONIN, sólo que esta vez no se debe introducir la secuencia de escape completa en la memoria. Introdúzcanse sólo los caracteres hex 1B y 4F. Asegúrense de que la rutina CONIN no coloca otra cadena en lugar de esta secuencia de casi escape.

Esta comprobación imita únicamente los resultados de las introducciones manuales de una secuencia de escape. No se pueden pulsar las teclas de un terminal a velocidad suficiente como para obtener los tres caracteres en la memoria intermedia de entrada dentro del tiempo permitido por el centinela.

Lista de control de salida de caracteres *¿Puede el controlador enviar un carácter?*

La opción CONOUT de la base de comprobación llama a CONIN en primer lugar para obtener un carácter. Para empezar, habrá que querer utilizar el DDT para fijar en el registro C algún carácter gráfico ASCII, tal como 41H (A), y transferir el control directamente a CONOUT. Compruébese que éste lee el estado de la USART, espera el valor de salida preparada y envía entonces los datos al puerto de datos. Obsérvese que la base de comprobación enviará todos los caracteres que esperan en la memoria intermedia de entrada (o entrada forzada) cuando se selecciona su opción CONOUT. Esto es muy conveniente para el control avanzado de los controladores —para comprobaciones iniciales puede quererse modificar la base de comprobación para hacer sólo una llamada a CONIN y a CONOUT y entonces volver al principio del bucle de la base de comprobación.

¿Suspende el controlador la salida cuando el indicador de control de protocolo indica que la salida ha de suspenderse?

Utilizando el DDT, colóquese el byte de estado de la tabla de dispositivo para activar el protocolo de salida XON/XOFF. Introdúzcase entonces un carácter XOFF y confírmese que el bit de salida suspendida del byte de estado está fijado. Envíese un solo carácter y, utilizando el DDT, confírmese que el controlador se quedará en un bucle de estado esperando que el bit de salida suspendida sea borrado. Bórrese el bit utilizando el DDT y compruébese que el carácter se ha enviado correctamente.

Cuando se utiliza el protocolo ETX/ACK, ¿envía el controlador un ETX después de que ha sido enviado el número especificado de caracteres, indicando entonces que se ha suspendido la salida?

Con fines de depuración, altérese el valor de la cuenta de mensaje ETX de la tabla de dispositivos a tres bytes. Envíense entonces tres

bytes de datos por medio de CONOUT. Compruébese que el controlador envía un carácter ETX (03H) después de que los tres bytes hayan sido enviados y que el indicador de salida suspendida del byte de estado haya sido fijado.

Introdúzcase entonces un carácter ACK (06H). Compruébese que este carácter no esté almacenado en la memoria intermedia de entrada y que el indicador de salida suspendida se ha borrado.

¿Reconoce y envía el controlador secuencias de escape?

Introdúzcase un ESCAPE, "t" (1BH, 74H), en la memoria intermedia de entrada. Envíense entonces por medio de CONOUT. Utilizando el DDT, compruébese que la rutina CONOUT reconoce que una secuencia de escape está siendo enviada y que selecciona la rutina de procesamiento correcta. En este caso, el apuntador de entrada forzada deberá ponerse apuntando a la hora del día en ASCII del bloque de configuración.

¿Funciona correctamente cada uno de los procesadores de secuencias de escape? ¿Se pueden fijar la hora y la fecha en valores específicos utilizando secuencias de escape?

Repítase la prueba anterior utilizando todas las demás secuencias de escape para asegurarse de que pueden ser reconocidas y que funcionan correctamente.

Rutinas de reloj de tiempo real

Un programa de base de comprobación separado, que se muestra en la figura 10-5, se utiliza para comprobar estas rutinas. Llama a la rutina de servicio de interrupción directamente para simular un "tic-tac" del reloj de tiempo real, y compone entonces la hora del día en ASCII en la consola.

Como puede verse, la base de comprobación hace una llamada a la rutina de inicialización del paquete de programas de depuración, DB$Init, y entonces utiliza un RTS 6 para generar "tic-tacs" de reloj falseados.

Hay una instrucción JMP en la base de comprobación que omite una llamada a Set$Watchdog. Trasládese este JMP bien editándolo o utilizando el DDT para cambiarlo por NO OPERATION (NOP, 00H) cuando se esté dispuesto a comprobar las rutinas de centinela.

Lista de control para comprobación del reloj de tiempo real *¿Está funcionando el reloj?*

Utilizando el DDT, escudríñese a través de la lógica de la rutina de servicio de interrupción. Compruébese que los segundos estén actualizándose.

```
                        ;       Base de test para controlador de reloj de tiempo real en el BIOS
                        ;
                        ;       El fichero fuente completo consiste en tres componentes:
                        ;
                        ;               1. El programa propiamente dicho que se muestra aquí.
                        ;               2. Los controladores de E/S de caracteres destinados al BIOS.
                        ;               3. El paquete de programas de depuración mostrado en la figura 10-2.
                        ;
FFFF =          TRUE    EQU     OFFFFH
0000 =          FALSE   EQU     NOT TRUE

FFFF =          DEBUG   EQU     TRUE            ;Para ensamblaje condicional de las
                                                ;  instrucciones RST en lugar de las IN
                                                ;  y OUT en los controladores
0030 =          RST6    EQU     30H             ;Utilización de RST 6 para simular un pulso de reloj

0100                    ORG     100H
                START:
0100 318B01             LXI     SP,Test$Stack   ;Utilización de una pila local
0103 CD8B01             CALL    DB$Init         ;Inicialización del paquete de depuración JMP
0106 3EC3               MVI     A,JMP           ;Fijación de la dirección de JMP de RST 6
0108 323000             STA     RST6
010B 218B01             LXI     H,RTC$Interrupt ;Activar dirección de JMP RST 6
010E 223100             SHLD    RST6 + 1

0111 C31D01             JMP     Testbed$Loop    ;<=== Quitar este JMP cuando se esté preparado para
                                                ;      comprobación de las rutinas de temporizador-centinela

0114 013200             LXI     B,50            ;50 pulsos antes de dar tiempo excedido
0117 214201             LXI     H,WD$Timeout    ;Dirección para transferencia
011A CD8B01             CALL    Set$Watchdog    ;Activación de temporizador-centinela
                        ;
                        ;
                        ;       Realizar entradas repetidas a la rutina de interrupción RTC
                        ;       para asegurar que el reloj está actualizado correctamente
                        ;
                        Testbed$Loop:
011D 3EAA               MVI     A,0AAH          ;Fijación de registros a una configuración conocida
011F 01CCBB             LXI     B,0BBCCH
0122 11EEDD             LXI     D,0DDEEH
0125 2111FF             LXI     H,0FF11H
0128 F7                 RST     6               ;Simulación de reloj de interrupción

0129 CD8B01             CALL    DB$MSGI         ;Visualización de mensaje en-línea
012C 436C6F636B         DB      'Clock =',0

0134 218B01             LXI     H,Time$In$ASCII ;Obtención de dirección de reloj en controlador
0137 CD8B01             CALL    DB$MSG          ;Visualización de valor de reloj en curso
                                                ; (Nota: Time$In$ASCII contiene
                                                ;  ya un salto de línea)
013A CD8B01             CALL    DB$MSGI         ;Visualización de mensaje en-línea
013D 0D00               DB      0DH,0           ;Retorno de carro

013F C31D01             JMP     Testbed$Loop
                        ;
                        ;       El control llega a este punto cuando el temporizador-centinela
                        ;       da tiempo excedido
                        WD$Timeout:
0142 CD8B01             CALL    DB$MSGI
0145 0D0A576174         DB      0DH,0AH,'Watchdog timed out',0
015A C9                 RET                     ;Retorno a la rutina centinela
                        ;
015B 9999999999         DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
016B 9999999999         DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
017B 9999999999         DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
                        Test$Stack:
                        ;
                        ;       Rutinas ficticias de aquellas mostradas en otras figuras
                        ;
                        ;       Rutinas del BIOS (Figura 8-10)
                        ;
                        RTC$Interrupt:          ;Rutina de servicio de interrupción para pulso de reloj
                        Set$Watchdog:           ;Fijación de temporizador-centinela
                        Time$In$ASCII:          ;Cadena ASCII de HH:MM:SS, LF, 0
                        ;
                        ;       Rutinas de depuración (Figura 10-2)
                        ;
                        DB$Init:                ;Inicialización de la depuración
                        DB$MSGI:                ;Visualización de mensaje en-línea
                        DB$MSG:                 ;Visualización de mensaje
```

Figura 10-5. Base de comprobación para el controlador de reloj de tiempo real del BIOS.

¿Se transfieren correctamente las horas, los minutos y los segundos?

Déjese correr a toda velocidad el programa de la base de comprobación. Debería verse la hora actualizándose en la pantalla de la consola —aunque puede ser actualizada mucho más rápidamente que en tiempo real.

Utilícese el DDT para fijar los minutos a 58 y entonces déjese correr de nuevo el reloj. ¿Señala la hora correctamente y pone los minutos a 00? Entonces fíjense las horas a 11 y los minutos a 58 y déjese correr el reloj. ¿Se convierten los minutos en horas y se ponen éstas de nuevo a 0?

Repítanse estas pruebas con las constantes de actualización del reloj fijadas para formato de 24 horas.

¿La rutina de servicio de interrupción de reloj almacena de nuevo los registros correctamente?

Utilizando el DDT, compruébese que los registros estén fijados aun correctamente al retorno de la rutina de servicio de interrupción del reloj.

¿Cuánta carga impone la rutina de servicio en la pila de preinterrupción?

Compruébese el "indicador de nivel" de los valores prefijados que quedan en el área de pila de la base de comprobación para ver cuánta carga impone a la pila la rutina de servicio de interrupción.

¿Puede fijarse el temporizador-centinela a un valor distinto a cero? ¿Puede volverse a poner a cero?

Utilizando la segunda parte de la base de comprobación, llámese a la rutina Set$Watchdog y compruébese entonces la ejecución de la base de comprobación cuando el cronómetro centinela está funcionando. Compruébese que los registros y el apuntador de pila están situados correctamente cuando se transfiere el control a la rutina de tiempo excedido. Compruébese también que se devuelve el control adecuadamente desde esta rutina y desde la rutina de servicio de interrupción.

Controladores de disco

Sólo es factible comprobar los controladores de disco de bajo nivel aislados de un BIOS real, porque la interfase del BDOS con el programa de desbloqueo es muy difícil de simular. La base de comprobación que se muestra en la figura 10-6 sirve únicamente para ganar tiempo. No comprueba la interfase de las subrutinas. Utilícese el DDT para fijar los números de disco, pista y sector y entonces compruébense las llamadas en SELDSK, SETTRK, SETSEC, SETDMA y las rutinas de lectura/escritura.

```
                ;       Base de test para controladores de E/S de disco en el BIOS
                ;
                ;       El fichero fuente completo consiste en tres componentes:
                ;
                ;               1. El programa propiamente dicho que se muestra aquí.
                ;               2. Los controladores de E/S de caracteres destinados al BIOS.
                ;               3. El paquete de programas de depuración mostrado en la figura 10-2
                ;
FFFF =          TRUE    EQU     0FFFFH
0000 =          FALSE   EQU     NOT TRUE

FFFF =          DEBUG   EQU     TRUE            ;Para ensamblaje condicional de las
                                                ;  instrucciones RST en lugar de las IN
                                                ;  y OUT en los controladores
0100                    ORG     100H
                START:
0100 314704             LXI     SP,Test$Stack   ;Utilización de una pila local
0103 CD4704             CALL    DB$Init         ;Inicialización del paquete de depuración
                ;
                ;       Realización de llamadas a SELDSK, SETTRK, SETSEC y SETDMA
                ;       y después a una rutina de lectura o escritura
                ;
                Testbed$Loop:
0106 314704             LXI     SP,Test$Stack   ;Utilización de pila local

0109 3A1202             LDA     Logical$Disk    ;Fijación para llamada a SELDSK
010C 4F                 MOV     C,A
010D CD4704             CALL    SELDSK

0110 CD4704             CALL    DB$Display      ;Visualización de valor de retorno en HL
0113 14                 DB      DB$HL
0114 53454C4453         DB      'SELDSK returned',0

0124 223201             SHLD    DPH$Start       ;Activación para visualizar cabecera de parámetros de disco
0127 111000             LXI     D,16            ;Cálculo de dirección de finalización
012A 19                 DAD     D
012B 223401             SHLD    DPH$End         ;Almacenamiento en llamada de depuración

012E CD4704             CALL    DB$Display      ;Visualización de DPH
0131 18                 DB      DB$M            ;Memoria
                DPH$Start:
0132 0000               DW      0
                DPH$End:
0134 0000               DW      0
0136 53656C6563         DB      'Selected DPH',0

0143 2A1302             LHLD    Track           ;Llamada SETTRK
0146 E5                 PUSH    H
0147 C1                 POP     B               ;SETTRK necesita pista en BC
0148 CD4704             CALL    SETTRK

014B 3A1502             LDA     Sector          ;Llamada a SETSEC
014E 4F                 MOV     C,A             ;SETSEC necesita sector en C
014F CD4704             CALL    SETSEC

0152 011702             LXI     B,Test$Buffer   ;Fijación de dirección de DMA
0155 CD4704             CALL    SETDMA
0158 3A1602             LDA     Write$Disk      ;Comprobación de lectura o escritura
015B B7                 ORA     A
015C C2D101             JNZ     Test$Write

015F CD4704             CALL    Read$No$Deblock;*** o Read$Physical dependiendo del controlador
                                                ;*** que se está comprobando
0162 CD4704             CALL    DB$Display      ;Visualización de código de retorno
0165 02                 DB      DB$A
0166 5465737420         DB      'Test Read returned',0

0179 CD0102             CALL    Check$Ripple    ;Comprobación de configuración correcta en mem. int.
017C CA0601             JZ      Testbed$Loop    ;Sí, correcta

017F CD4704             CALL    DB$MSGI         ;Indicar problemas
0182 14                 DB      DB$HL           ;Visualización de HL (apunta al byte incorrecto)
0183 526970706C         DB      'Ripple pattern incorrect. HL -> failure.',0

01AC CD4704             CALL    DB$Display      ;Visualización de memoria intermedia de test
01AF CD1800             CALL    DB$M            ;Memoria
01B2 1702               DW      Test$Buffer
```

Figura 10-6. Base de comprobación para controladores de entrada/salida de disco en el BIOS.

```
01B4 0002              DW      Test$Buffer$Size
01B6 436F6E7465        DB      'Contents of Test$Buffer',0

01CE C30601            JMP     Testbed$Loop

            Test$Write:
01D1 CDF201            CALL    Fill$Ripple      ;Llenar la memoria intermedia de test con
01D4 CD4704            CALL    Write$No$Deblock ;*** o con Write$Physical dependiendo de cuáles
                                                ;*** sean las unidades comprobadas

01D7 CD4704            CALL    DB$Display       ;Visualización código de retorno
01DA 02                DB      DB$A
01DB 5465737420        DB      'Test Write returned',0

01EF C30601            JMP     Testbed$Loop

            Fill$Ripple:                        ;Llenado de la memoria intermedia de test con
                                                ;  una configuración de bits formada poniendo
                                                ;  en cada byte los 8 bits menos significativos
                                                ;  de la dirección de los bytes
01F2 010002            LXI     B,Test$Buffer$Size
01F5 211702            LXI     H,Test$Buffer
            FR$Loop:
01F8 75                MOV     M,L              ;Fijar valor de la configuración en la memoria intermedia
01F9 23                INX     H                ;Actualización de apuntador a memoria intermedia
01FA 0B                DCX     B                ;Decrementar contador
01FB 79                MOV     A,C              ;Comprobación de contador a cero
01FC B0                ORA     B
01FD C2F801            JNZ     FR$Loop          ;Repetición hasta cero
0200 C9                RET
            ;
            Check$Ripple:                       ;Comprobación de si la memoria intermedia se ha
                                                ;  llenado con la configuración correcta.
                                                ;  Retorno con estado 0 si es cierto,
                                                ;  estado distinto de cero si es incorrecta.
                                                ;  HL apunta al byte incorrecto
                                                ;  (que será = L)
0201 010002            LXI     B,Test$Buffer$Size
0204 211702            LXI     H,Test$Buffer
            CR$Loop:
0207 7D                MOV     A,L              ;Obtención valor correcto
0208 BE                CMP     M                ;Comparación con contenido memoria intermedia
0209 C0                RNZ                      ;No coincidencia, distinto de cero ya indicado
020A 23                INX     H                ;Actualización apuntador a memoria intermedia
020B 0B                DCX     B                ;Decrementar contador
020C 79                MOV     A,C              ;Comprobación contador a cero
020D B0                ORA     B
020E C20702            JNZ     CR$Loop          ;Repetición hasta 0
0211 C9                RET                      ;Indicador de cero estará activo
            ;
            ;       Testbed variables
            ;
0212 00     Logical$Disk:  DB      0        ;A = 0, B = 1,...
0213 0000   Track:         DW      0        ;Número de pista
0215 00     Sector:        DB      0        ;Número de sector
0216 00     Write$Disk:    DB      0        ;NZ a escribir en disco
            ;
0200 =      Test$Buffer$Size       EQU     512     ;<=== Distinto del necesario
0217        Test$Buffer:   DS      Test$Buffer$Size
            ;
0417 9999999999        DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
0427 9999999999        DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
0437 9999999999        DW      9999H,9999H,9999H,9999H,9999H,9999H,9999H,9999H
            Test$Stack:
            ;
            ;       Rutinas ficticias de las mostradas en otras figuras
            ;
            ;       Rutinas del BIOS (figura 8-10)
            ;
            SELDSK:                 ;Selección de disco lógico
            SETTRK:                 ;Fijación número de pista
            SETSEC:                 ;Fijación número de sector
            SETDMA:                 ;Fijación de dirección de DMA
            Read$No$Deblock:        ;Controlador rutinas de lectura
            Read$Physical:
            Write$No$Deblock:       ;Controlador rutinas de escritura
            Write$Physical:
```

Figura 10-6. (Continuación.)

```
                ;
                ;       Rutinas de depuración (figura 10-2)
                ;
                DB$Init:                ;Inicialización de depuración
                DB$MSGI:                ;Visualización de mensaje en-línea
                DB$Display:             ;Rutina principal de visualización de depuración
0002 =          DB$A    EQU     02      ;Visualización de códigos para DB$Display
0014 =          DB$HL   EQU     20
0018 =          DB$M    EQU     24
```

Figura 10-6. (Continuación.)

A menos de que se tenga el mismo controlador de disco en el sistema de desarrollo como se tiene en la máquina final, habrá que utilizar el sistema de entrada/salida falseado descrito anteriormente en este capítulo, mejor que intentar leer y escribir en discos reales.

Como puede verse, la base de comprobación, después de inicializar el paquete de depuración, realiza llamadas a SELDSK, SETTRK, SETSEC y SETDMA. Entonces llama una rutina de lectura o de escritura de bajo nivel. La rutina de bajo nivel llamada depende del controlador que se desee depurar. Para el controlador de disquetes estándar que se muestra en la figura 8-10, utilícese ReadNoDeblock y WriteNoDeblock. Para los disquetes de 5 1/4 pulgadas, utilícese Read$Physical y Write$Physical. Habrá de utilizarse el DDT para fijar algunas de las variables requeridas por los controladores de bajo nivel que normalmente se fijan por medio del programa de desbloqueo.

Antes de lanzar la llamada de escritura, la base de comprobación llena la memoria intermedia del disco con una configuración conocida. Esta configuración se comprueba al volver de una operación de lectura.

Tanto para lectura como para escritura la base de comprobación muestra los contenidos del registro A. Si se ha añadido el manejo mejorado de errores de disco que se describía en el capítulo anterior, el valor de retorno de A deberá ser *siempre* cero.

Lista de comprobación de controladores de disco *¿Devuelve SELDSK la dirección correcta y fija las variables de sistema requeridas?*

Compruébese que la dirección de cabecera de parámetros del disco correcta ha vuelto para los discos lógicos legítimos. Compruébese también que devuelve una dirección 0000H para los discos ilegales.

Compruébese que cualquier proceso rutinario, tal como fijación del tipo de disco y requerimientos de desbloqueo de bytes extra de los bloques de parámetros del disco, se realiza correctamente.

¿Funcionan correctamente los procesos SETTRK y SETSEC?

Utilizando el DDT, compruébese que las variables correctas están fijadas en los valores específicos.

¿Lee el controlador correctamente en el directorio de sectores de repuesto?

Colóquese para ejecutar lectura física y, utilizando el DDT, examínese la lógica del punto de entrada READ. Compruébese que el directorio de sectores de repuesto esté cargado en la memoria intermedia adecuada. Si se está utilizando entrada/salida falseada, utilícese el DDT para añadir un directorio típico de sectores de repuesto con dos o tres sectores.

¿Produce el controlador el sector de repuesto correcto en lugar de uno malo?

Continuando con la operación física de lectura, compruébese que para pistas/sectores "buenos", la lógica de repuesto de sectores devuelve los números originales de pista y sector, y para pistas/sectores "malos", sustituye éstos por pistas y sectores de repuesto correctos. Si se está utilizando saltos de sector, compruébese que se salta el número correcto de sectores.

¿Puede leerse un sector desde el disco?

Continuando adelante con la lectura física, compruébese que se lee el sector correcto del disco y pista específicos. Si se están utilizando entradas/salidas reales (como opuesto a falsearlas), la "configuración" fijada por la base de comprobación puede ser usada, o puede llenarse el área de memoria intermedia del disco con alguna configuración conocida (utilizando la orden F del DDT) con lo que puede decirse si se lee algún dato.

Hay que asegurarse de que no haya ningún disco o disquete en la computadora no protegidos contra escritura —pues se puede escribir inadvertidamente en un disco más que leerlo durante los primeros pasos de la comprobación.

¿Puede escribirse un sector en el disco?

Utilizando el DDT, fíjese para escribir en un disco, pista y sector particulares. Quítese cualquier protección de escritura que se haya puesto en este disco durante comprobaciones anteriores. Se puede utilizar bien la configuración dada en la base de comprobación o llenar el área de memoria intermedia del disco con una configuración distintiva. Escríbanse estos datos en el disco, llénese el área del disco con una configuración *diferente* y léase en el sector que se escribió. Compruébese que la memoria intermedia del disco se cambia a la configuración escrita en el disco.

¿Compone el controlador mensajes de error correctamente?

Mejor que dañar un disquete deliberadamente para crear errores, utilícese el DDT para el sabotaje temporal de la lógica del controlador del disco. Hágase que devuelva cada uno de los códigos de error

posibles por turno, comprobando cada vez que ha sido lanzado el mensaje de error correcto.

Para cada condición de error por turno, compruébese que el controlador de disco realiza la acción de recuperación correcta, incluyendo la interacción con el usuario y ofreciendo la opción de reintentarlo, ignorar el error, o la de abortar el programa.

Comprobación en vivo de un nuevo BIOS

Suponiendo que los controladores hayan pasado todos los controles señalados anteriormente, todo está preparado para montar todas las piezas del BIOS y construir una imagen del CP/M.

Para la comprobación inicial, hay que desconectar el reloj de tiempo real y utilizar entradas/salidas sencillas para el controlador de la consola, si es posible. Es importante obtener *algo* que funcione lo más pronto posible, y es más fácil conseguirlo sin los posibles efectos secundarios de las interrupciones.

Prepárese un listado completo del BIOS y planifíquese perder al menos una hora comprobándolo. Realícese una pasada en seco a través de la consola y del controlador de disco —si existen errores graves en estos dos controladores, el CP/M no se pondrá en marcha—. Recuérdese que una vez se ha ejecutado el arranque en frío del BIOS y se ha transferido el control al CCP, se pedirá al BDOS que entre en el disco del sistema, y esto incluye la lectura del directorio del disco.

Obsérvese especialmente el controlar algunas de las estructuras de datos más importantes. Hay que asegurarse de que todo está en un lugar razonable de la memoria; por ejemplo, si la última dirección utilizada por el BIOS es mayor de 0FFFFH, se necesitará quitar la imagen entera del CP/M de la memoria.

Constrúyase entonces un disco de sistema, cárguese en la máquina y púlsese el botón de RESET. Se verá la señal de arranque, después el BIOS y tras una pausa de un segundo aproximadamente la indicación A> (o 0A> si se ha incluido el dispositivo especial que añade el CCP).

Si se ven ambos mensajes de "funcionamiento" pero no se obtiene la indicación A>, probablemente la causa del problema se encuentra en los controladores de disco. De forma alternativa, el área de directorio del disco puede estar llena de datos cualesquiera mejor que 0E5H.

Si no se puede ver lo que está equivocado en el sistema, puede intentarse falsear los controladores de disco para que devuelvan un bloque de 128 bytes 0E5H en cada operación de lectura. El CCP lanzará entonces el mensaje de "funcionando".

Una vez que ha aparecido la indicación A>, se puede proceder a la

comprobación del sistema. Se empieza comprobando si funciona la lógica de autocarga. Mecanografíese un CONTROL-C. Habrá una ligera pausa y la indicación A> deberá aparecer de nuevo.

A continuación se comprueba que se pueda leer el directorio del disco utilizando la orden DIR. Si se tiene un directorio vacío, se obtendrá una respuesta NO FILE. Si se obtienen caracteres extraños en su lugar, o bien se ha olvidado la inicialización del área del directorio o el bloque de parámetros del disco está dirigiendo el CP/M a la parte errónea del disco para el directorio de ficheros. Si cae el sistema, hay problemas con el controlador de disco.

Compruébese que se puede escribir en el disco introduciendo la orden SAVE 1 TEST. Utilícese entonces la orden DIR para confirmar que el fichero TEST se muestra en el directorio de ficheros. Si ocurre así, utilícese la orden ERA, ERA TEST y hágase otra orden DIR para confirmar que se ha borrado TEST.

Si TEST no aparece en el disco o no puede ser borrado, entonces se tiene un problema con la rutina WRITE del controlador de disco.

Póngase un disquete estándar del CP/M en la unidad B y utilícese la orden DIR para comprobar que se puede acceder al controlador y visualizar un directorio de disco. Si es así, entonces cárguese desde él la utilidad DDT y sálgase desde él utilizando una orden G0 (G, cero). Esto comprobará mejor si los controladores de disco están funcionando correctamente.

Para comprobar la lógica de desbloqueo (si se están usando discos que requieran desbloqueo), utilícese la orden:

```
PIP A:=B:*.*[V]
```

Esta copia todos los ficheros de la unidad B en la unidad A, utilizando la opción de verificación. Es una prueba particularmente buena del sistema, y si se tienen problemas con los controladores de disco de alto nivel y con el programa de desbloqueo, se obtendrá un mensaje de verificación de error del PIP. También puede obtenerse este mensaje cuando se tienen problemas de hardware con la memoria de la computadora, por lo que hay que dejar correr una prueba de memoria si no se encuentra nada equivocado en el algoritmo de desbloqueo.

Para comprobar totalmente el programa de desbloqueo, se necesita utilizar el PIP para copiar un fichero de texto mayor que el total de memoria disponible. Entonces se puede necesitar crear un fichero grande de texto utilizando un editor de texto exclusivamente para proporcionar datos de control al PIP.

Con el controlador de disco funcionando correctamente, reconstruir el sistema con el reloj de tiempo real activado. Arránquese el nuevo sistema y compruébese que la hora del día del ASCII esté siendo actualizada en el bloque de configuración; utilícese el DDT para inspeccionarlo en la

memoria. Póngase el reloj en la hora en curso, déjese andar durante cinco minutos y véase si se encuentra todavía correcto. Quizá haya que ajustar una de las constantes de inicialización de tiempo para el dispositivo que procura las interrupciones periódicas del reloj.

Reconstrúyase el sistema ahora una vez más, esta vez con el controlador de interrupciones de entrada de consola real y las rutinas de salida real de la consola. Compruébese que el sistema funciona adecuadamente y que la cadena de puesta en marcha de la entrada forzada inicial aparece en la consola.

Compruébese que cuando se mecanografían caracteres en el teclado se visualizan según se van mecanografiando. Si no, puede haber problemas bien en la rutina CONIN o en la CONOUT. Experimentalmente mecanografíense caracteres suficientes para llenar la memoria intermedia de entrada. Si empieza a sonar la campana del terminal, la rutina de servicio de interrupción probablemente no es la culpable. Compruébese de nuevo la rutina CONOUT.

Compruébese que la lógica de decodificación de la tecla de función trabaje correctamente. Con la indicación A> compuesta, púlsese una tecla de función. El controlador CONIN deberá inyectar la cadena de tecla de función correcta y ésta deberá aparecer en el terminal. Por ejemplo, con el BIOS de la figura 8-11, pulsando PF1 en el terminal VT-100, se producirá la siguiente presentación:

```
A>Function Key1
Function?
A>
```

El CCP no reconoce "Function" como un nombre legítimo de orden, ni existe un fichero COM —de ahí el interrogante.

Utilizando el DDT, escríbase un pequeño programa que envíe ESCAPE, "t" a la consola y compruébese que la cadena de hora del día ASCII aparece en la consola. Esto prueba que la secuencia de escape ha sido reconocida.

Funciones de biblioteca
Lectura o escritura de un disco utilizando el BIOS
Programas de utilidad que mejoran el CP/M estándar
Programas de utilidad para el BIOS mejorado

11

Programas de utilidad adicionales

Este capítulo contiene el programa fuente ilustrado de varios programas de utilidad muy prácticos. Se incluyen dos grupos de dichos programas —los que suplementan los programas de utilidad estándar de Digital Research y los que trabajan junto con características de las que se muestran en el BIOS mejorado (figura 8-10).

Para evitar detalles innecesarios, los programas que se presentan en este capítulo están todos escritos en lenguaje C. Este es un buen lenguaje para usarlo con este fin porque puede mostrar la totalidad de la lógica de un programa sin la confusión de detalles corrientes en los lenguajes de ensamblaje.

Con el fin de volver a utilizar tanto programa fuente como sea posible, este capítulo incluye una "biblioteca" de todas las funciones de propósito general de C que pueden llamarse desde dentro de cualquiera de los programas de utilidad. Este fichero, denominado "LIBRARY.C", se muestra en la figura 11-1. Una vez el programa ha sido compilado, las funciones necesarias de la biblioteca pueden enlazarse con la salida binaria de utilidades para formar el fichero ".COM".

```
/* Biblioteca de funciones utilizadas frecuentemente */

#include <LIBRARY.H>     /* Definiciones y estructuras estándar */

/*       Acceso al bloque de configuración */

/*=============================================================*/
char
*get_cba(code)     /* Obtención de la dirección bloque de configuración */
/*=============================================================*/
/* Esta función realiza una llamada a una entrada "privada" al vector
   de saltos del BIOS para devolver la dirección de un dato específico
   en el BIOS. El programa indica cuál es el dato requerido.
   Cada uno de los programas que utiliza esta función puede realizar
   una llamada directa al BIOS utilizando la función biosh() que proporciona
   el BDS C. Esta función tiene un punto común al cual puede añadirse
   el programa de depuración para visualizar la dirección de vuelta. */

/* Parámetros de entrada */
int code;         /* Programa que especifica el objeto
                     cuya dirección se requiere */
/* Parámetros de salida
   Dirección devuelta por la rutina del BIOS */

{
char *retval;             /* Valor devuelto por el BIOS */

        retval = biosh(CBGADDR,code);
 /* printf("\nget_cba : code %d address %4x",code,retval); */
        return retval;
} /* Final de get_cba(programa) */


/*     Funciones de manipulación de caracteres       */

/*=============================================================*/
strscn(string,key)              /* Revisión de cadenas */
/*=============================================================*/
/* Esta función examina una cadena de caracteres terminada por 00
   buscando en ella una clave. Si se encuentra la clave
   en la cadena, la función devuelve un apuntador a él. En caso contrario
   devuelve un valor 0. */
/* Parámetros de entrada */
char *string;             /* Cadena a examinar */
char *key;                /* Clave a buscar */

/* Parámetros de salida
   Apuntador a clave, de cadena en cadena examinada, o si no
   encuentra la clave
*/

{
while (*string)           /* Para todos los caracteres no nulos en la cadena */
        {
        if ((*string == *key) &&         /* Coincidencia con primer carácter */
            (sstrcmp(string,key) == 0)   /* Configurar con subcadena
                                            comparación con el resto */
           )
                return string;           /* Coincidencia con subcadena,
                                            retorno de apuntador */
        string++;                        /* Desplazamiento a carácter siguiente de la cadena */
        }
return 0;                                /* Indicación de coincidencia encontrada */
} /* Final de strscn */


/*=============================================================*/
ustrcmp(string1,string2)     /* Comparación de cadenas en mayúsculas */
/*=============================================================*/
/* Esta función es similar a la función normal strcmp;
   difiere sólo en que los caracteres son comparados
   si son caracteres en mayúsculas -- las cadenas permanecen
   inalteradas. */
```

a

b

c

Figura 11-1. Funciones utilizadas normalmente, LIBRARY.C, en lenguaje C.

```
/* Parámetros de entrada */
char *string1;              /* Apuntador a primera cadena */
char *string2;              /* Apuntador a segunda cadena */

/* Parámetros de salida
   0 - si cadena 1 = cadena 2
   negativo si cadena 1 > cadena 2
   positivo si cadena 1 < cadena 2
*/

{
int count;                  /* Utilizado para acceso a los caracteres en las dos cadenas */

count = 0;                  /* Comienzo en el primer carácter de las dos */

        /* Mientras los caracteres de la cadena 1 son no nulos  y
           coinciden con sus correspondientes en la cadena 2. */
while (string1[count] == string2[count])
        {
        if (string1[++count] == '\0')    /* Ultimo carácter en cadena 1 */
                return 0;                /* Indicador de igualdad */
        }
return string2[count] - string1[count]; /* Comparación de caracteres */

} /* Final de sstrcmp */
```

c

```
/*=============================================================*/
sstrcmp(string,substring)               /* Comparación de subcadenas */
/*=============================================================*/
/* Esta función compara dos cadenas. La primera de ellas no debe
   terminar necesariamente por 00. La segunda debe terminar por 00.
   Es similar a la función strcmp excepto en que la longitud
   de la subcadena controla cuantos caracteres han sido comparados. */

/* Parámetros de entrada */
char *string;               /* Apuntador a cadena principal */
char *substring;            /* Apuntador a subcadena */

/* Parámetros de salida
   0 - la subcadena coincide con los caracteres correspondientes en la cadena
   negativo si caracteres en cadena > caracteres en subcadena
   positivo si caracteres en cadena < caracteres en subcadena
*/

{
int count;        /* Utilizado para acceso a caracteres en cadena y subcadena */

count = 0;        /* Comienzo con el primer carácter de cada uno */

        /* Mientras los caracteres de la subcadena son no nulos y
           coinciden con los correspondientes en la cadena. */
while (string[count] == substring[count])
        {
        if (substring[++count] == '\0') /* Ultimo carácter en subcadena */
                return 0;               /* Indicación de igualdad */
        }
return substring[count] - string[count];          /* Comparación de caracteres */

} /* Final sstrcmp */
```

d

```
/*=============================================================*/
usstrcmp(string,substring)      /* Comparación de subcadenas en mayúsculas */
/*=============================================================*/
/* Esta función compara dos cadenas. La primera no necesita
   terminar en 00. La segunda subcadena debe terminar en 00.
   Es similar a la comparación de subcadenas anterior excepto
   que todos los caracteres son mayúsculas. */

/* Parámetros de entrada */
char *string;               /* Apuntador a cadena principal */
char *substring;            /* Apuntador a subcadena */

/* Parámetros de salida
   0 -- subcadena coincidente con caracteres correspondientes en cadena
```

e

Figura 11-1. (Continuación.)

```
    negativo si carácter en cadena > carácter en subcadena
    positivo si carácter en cadena < carácter en subcadena
*/

{
int count;        /* Utilizado para acceso a caracteres en cadena y subcadena */

count = 0;        /* Comienzo con el primer carácter de cada uno */

        /* Mientras los caracteres de la subcadena son no nulos y
           coinciden con los correspondientes en la cadena. */
while (toupper(string[count]) == toupper(substring[count]))
        {
        if (substring[++count] == '\0')  /* Ultimo carácter en subcadena */
                return 0;                /* Indicación de igualdad */
        }
return substring[count] - string[count];         /* Comparación de caracteres */
} /* Final de usstrcmp */


/*************************************************************/
comp_fname(scb,name)              /* Comparación de nombres de fichero */
/*************************************************************/
/* Esta función compara nombres de fichero posiblemente ambiguos
   con el nombre en cadena de caracteres especificada. El número
   de bytes comparados viene determinado por el número de caracteres
   en la máscara.
   Esta función puede ser utilizada en la comparación de nombres y tipos,
   o, añadiendo un byte extra a la máscara, nombres, tipos y extensiones
   de fichero.
   Para las entradas al directorio debe anteponerse un byte en la
   máscara y utilizar la función para comprobar el número de
   usuario y el nombre, tipo y extensión del fichero.
   Nótese que "?" en el primer carácter de la máscara NO coincide
   con el valor 0xE5 (este valor se utiliza para indicar una entrada
   al directorio no activa). */

/* Parámetros de entrada */
struct _scb *scb;          /* Apuntador a bloque de control de búsqueda */
char *name;                /* Apuntador a nombre de fichero */

/* Parámetros de salida
   NAME_EQ si los nombres coinciden con la máscara
   NAME_LT si el nombre es menor que la máscara
   NAME_GT si el nombre es mayor que la máscara
   NAME_NE si el nombre no es igual que la máscara (pero el resultado
        es ambiguo debido a los comodines) en la máscara
*/

{
int count;                 /* Cuenta del número de caracteres procesados */
short ambiguous;           /* NZ cuando la máscara es ambigua */
char *mask;                /* Apuntador a los bytes al principio del SCB */

/* Fijación de apuntador a los caracteres al principio del bloque de
control de búsqueda */
mask = scb;
        /* Coincidencia ambigua en número de usuario, compara sólo
           usuarios 0-15 y no entradas no activas */
if (mask[0] == '?')
        {
        if (name[0] == 0xE5)
                return NAME_NE;  /* Indicación de desigualdad */
        }
else    /* Primer carácter de la máscara no es "?" */
        {
        if (mask[0] != name[0]) /* Número de usuario no coincidente */
                return NAME_NE; /* Indicación de desigualdad */
        }

/* No, comprobación del nombre (y, si la longitud es tal, la extensión) */
for (count = 1;                 /* Comienzo con el primer carácter del nombre */
     count <= scb -> scb_length;/* Para todos los caracteres requeridos */
     count++)                   /* Desplazamiento a carácter siguiente */
        {
        if (mask[count] == '?') /* Carácter comodín en la máscara */
```

Figura 11-1. (Continuación.)

```
                {
                ambiguous = 1;   /* Indicación de nombre ambiguo en la máscara */
                continue;        /* No se realizan comparaciones */
                }
        if (mask[count] != (name[count] & 0x7F))
                {                /* Máscara de caracteres distinto de caracteres FCB */
                if (ambiguous)   /* Si comodín anterior, indicación de NE */
                        return NAME_NE;
                else
                        /* Comparación de caracteres para determinar relación */
                        return (mask[count] > name[count] ?
                                NAME_LT : NAME_GT);
                }
        }
        /* Si el control llega a este punto, todos los caracteres de la máscara
        y nombre han sido procesados y o bien han sido comodines en la máscara
        o bien ha habido coincidencia. */
return NAME_EQ;                  /* Indicación de qué máscara y nombre son iguales */

} /* Final de comp_fname */
```

f

```
/*================================================================*/
conv_fname(fcb,fn)               /* Conversión de nombre de fichero para salida */
/*================================================================*/
/* Esta función convierte el contenido en bloque de control
   de ficheros en una cadena imprimible "D:FILENAME.TYP." */

/* Parámetros de entrada */
struct _fcb *fcb;                /* Apuntador a bloque de control de ficheros */
char *fn;                        /* Apuntador a área que recibirá el nombre */

{
        /* Si la especificación del disco en el FCB
           es 0, se utiliza el disco en curso */
*fn++ = (fcb -> fcb_disk) ? (fcb -> fcb_disk + ('A'-1)) :
        (bdos(GETDISK) + 'A');

*fn++ = ':';                          /* Insertar delimitador de identificador de disco */

movmem(&fcb -> fcb_fname,fn,8);       /* Transferencia de nombre de fichero */
fn += 8;                              /* Actualización de apuntador */
*fn++ = '.';                          /* Insertar delimitador nombre/tipo de fichero */
movmem(&fcb -> fcb_fname+8,fn,3);     /* Transferencia tipo de fichero */
*fn++ &= 0x7F;                        /* Eliminar bits de atributos */
*fn++ &= 0x7F;                        /* Eliminar bits de atributos */
*fn++ &= 0x7F;                        /* Eliminar bits de atributos */
*fn = '\0';                           /* Terminal */

} /* Final de conv_fname */
```

g

```
/*================================================================*/
conv_dfname(disk,dir,fn)         /* Conversión de un nombre de fichero del direct. para salida */
/*================================================================*/
/* Esta función convierte el contenido de un bloque de entrada del
   directorio de ficheros en una cadena imprimible "D.FILENAME.TYP" */

/* Parámetros de entrada */
short disk;                      /* Identificador de disco (A = 0, B = 1) */
struct _dir *dir;                /* Apuntador a bloque de control de fichero */
char *fn;                        /* Apuntador a área para recepción de nombre */

{
        /* Conversión de número de usuario e identificador de disco */
sprintf(fn,"%2d/%c:",dir -> de_userno,disk + 'A');
fn += 5;                         /* Actualización de apuntador a nombre de fichero */

movmem(&dir -> de_fname,fn,8);   /* Transferencia de tipo de fichero */
fn += 8;                         /* Actualización de apuntador */
*fn++ = '.';                     /* Insertar delimitador nombre/tipo de fichero */

movmem(&dir -> de_fname+8,fn,3); /* Transferencia tipo de fichero */
*fn++ &= 0x7F;                   /* Eliminar bits de atributos */
*fn++ &= 0x7F;                   /* Eliminar bits de atributos */
*fn++ &= 0x7F;                   /* Eliminar bits de atributos */
*fn = '\0';                      /* Terminal */
```

h

Figura 11-1. (Continuación.)

```
} /* Final de conv_dfname */                                                    h

/*==================================================================*/
get_nfn(amb_fname,next_fname)   /* Obtención nombre de fichero siguiente */
/*==================================================================*/
/* Esta función fija el FCB al "next_fname" para contener la entrada
   al directorio que coincide con el nombre de fichero ambiguo
   en "amb_fname".
   En la primera entrada para un nombre de fichero dado el bit más
   significativo del campo de disco del FCB debe fijarse a uno (esto causa la
   realización de una llamada a la "búsqueda del primer caso del BDOS". */

/* Parámetros de entrada */
struct _fcb *amb_fname;  /* Nombre de fichero ambiguo */
struct _fcb *next_fname; /* Primer byte debe tener a uno el bit más significativo
                            para entrada por primera vez */

/* Parámetros de salida
   0 = No más nombres encontrados
   1 = Encontrado nombre adicional (y fijado en next_fname)
*/

{
char bdos_func;          /* Fijación a búsqueda del primero o del siguiente */
char *pfname;            /* Apuntador a nombre de fichero en entrada al directorio */

        /* Inicialización a cero del final del FCB del fichero siguiente */
setmem(&next_fname -> fcb_extent,FCBSIZE-12,0);

bdos_func = SEARCHF;     /* Suposición de que realiza una búsqueda del primer caso */      i

if (!(next_fname -> fcb_disk & 0x80))    /* Si no la primera vez */
        {
                /* Búsqueda del primer caso con nombre anterior */
        srch_file(next_fname,SEARCHF);
        bdos_func = SEARCHN;             /* Entonces realizar búsqueda del siguiente */
        }
else    /* Primera vez */
        next_fname -> fcb_disk &= 0x7F; /* Puesta a cero indicador de primera vez */

        /* Refresco de next_fname desde nombre de fichero ambiguo */
           (move disk, name, type) */
movmem(amb_fname,next_fname,12);

        /* Si se trata de la primera vez, se lanza una llamada a búsqueda
           del primer caso, en contrario, a búsqueda del siguiente. "Srch_file"
           devuelve un apuntador a la entrada del directorio que coincide
           con el nombre de fichero ambiguo o 0 si no hay coincidencia */
if (!(pfname = srch_file(next_fname,bdos_func)) )
        {
        return 0;        /* Indicación de no coincidencia */
        }
        /* Transferir nombre y tipo de fichero */
movmem(pfname,&next_fname -> fcb_fname,11);
return 1;                /* Indicación de coincidencia encontrada */

} /*  Final de get_nfn  */

/*==================================================================*/
char *srch_file(fcb,bdos_code)   /* Búsqueda de fichero */
/*==================================================================*/
/* Esta función lanza una llamada a una de las dos funciones del BDOS,
   búsqueda del primer caso o búsqueda del siguiente. */

/* Parámetros de entrada */
struct _fcb *fcb;        /* Apuntador al bloque de control de fichero */                j
short bdos_code;         /* SEARCHF o SEARCHN */

/* Parámetros de salida
   0 = No coincidencia
   NZ = Apuntador a la entrada coincidente (en la memoria intermedia)
*/
```

Figura 11-1. (Continuación.)

```
{
unsigned r_code;          /* Devolución de un código desde la función de búsqueda
                             Este código es 255 para indicar no coincidencia a 0, 1, 2 ó 3
                             como ordinal de la entrada de 32 bytes en la memoria
                             intermedia con la que se ha encontrado coincidencia */
char *dir_entry;          /* Apuntador a entrada al directorio */

        /* El compilador C del BDS fija siempre la dirección de DMA
           del BDOS en la posición 0x80 */

r_code = bdos(bdos_code,fcb);    /* Lanzamiento de llamada al BDOS */
if (r_code == 255)               /* No coincidencia */
        return 0;

        /* Fijación de un apuntador a la entrada coincidente
           multiplicando el código devuelto por 128 y añadiéndolo
           a la dirección de la memoria intermedia (0x80) y añadiendo 1
           para apuntar al primer carácter del nombre */

return (r_code << 5) + 0x81;

}/* Final de srch_file */                                                    j


/*****************************************************************/
rd_disk(drb)              /* Lectura de disco (vía el BIOS) */
/*****************************************************************/
/* Esta función utiliza los parámetros fijados previamente
   en bloque requerido y ejecuta la lectura de disco
   usando directamente el BIOS. */

/* Parámetros de entrada */
struct _drb *drb;         /* Bloque de disco requerido (disco, pista, sector, memoria intermedia) */

/* Parámetros de salida
   0 = No existen datos disponibles
   1 = Existen datos disponibles
*/

{
if (!set_disk(drb))       /* Llamada a SELDSK, SETTRK, SETSEC */
        return 0;         /* Si SELDSK falla, indicar que no
                             hay datos disponibles */
if (bios(DREAD))          /* Ejecutar lectura */
        return 0;         /* Indicación de no existencia de datos disponibles si se recibe un error */

return 1;                 /* Indicación de datos disponibles */

} /*  Final de rd_disk  */                                                  k


/*****************************************************************/
wrt_disk(drb)             /* Escritura del disco (vía BIOS) */
/*****************************************************************/
/* Esta función utiliza los parámetros fijados previamente en el
   bloque de entrada requerido y ejecuta una escritura en
   disco usando directamente el BIOS. */

/* Parámetros de entrada */
struct _drb *drb;         /* Bloque de disco requerido (disco, pista, sector...) */

/* Parámetros de salida
   0 = Error durante la escritura
   1 = Datos escritos correctamente
*/

{
if (!set_disk(drb))       /* Llamada a SELDSK, SETTRK, SETSEC, SETDMA */
        return 0;         /* Si falla SELDSK, indicación de no escritura de datos */
if (bios(DWRITE))         /* Ejecución de escritura */
        return 0;         /* Indicación del error devuelto */

return 1;                 /* Indicación de datos escritos */

} /*  Final de wrt_disk  */                                                 l
```

Figura 11-1. (Continuación.)

```
/*=============================================================*/
short set_disk(drb)      /* Fijación de parámetros de disco */
/*=============================================================*/
/* Esta función fija las variables del BIOS anticipándose a una lectura
   o escritura de disco que sucederá a continuación. */

/* Parámetros de entrada */
struct _drb *drb;        /* Bloque de disco requerido (disco, pista, sector, memoria intermedia) */

/* Parámetros de salida
   0 = Disco no válido (no puede realizarse lectura/escritura)
   1 = BIOS activado para lectura/escritura
*/

{
        /* El sector en el bloque de disco requerido contiene un sector
           lógico. Si es necesario (según lo determine el valor
           en los parámetros cabecera de disco), debe de ser
           convertido en sector físico.
           NOTA: Skewtab se declara como un apuntador a un apuntador
           a un entero de un único byte. */
short **skewtab;         /* Skewtab -> cabecera de parámetros del disco -> tabla de desplaz. */
short phy_sec;           /* Sector físico */

        /* Llamada al punto de entrada SELDSK del BIOS. Si devuelve 0,
           el disco es no válido. En caso contrario devuelve un
           apuntador al apuntador a la tabla de desplazamiento */
if ( !(skewtab = biosh(SELDSK,drb -> dr_disk)) )
        return 0;                /*  Disco no válido  */

bios(SETTRK,drb -> dr_track);    /* Fijación de pista */

        /* Nótese que la función biosh sitúa en el registro BC el
           sector y un apuntador a la tabla de desplazamiento
           en los registros BC. Devuelve un valor en HL en la salida
           desde el BIOS */
phy_sec = biosh(SECTRN,drb -> dr_sector,*skewtab); /* Obtención de sector físico */
bios(SETSEC,phy_sec);            /* Fijación de sector */
bios(SETDMA,drb -> dr_buffer);   /* Fijación de dirección de memoria intermedia */

return 1;                /* Indicación de no existencia de irregularidades */

} /* Final de setp_disk */


/*      Funciones de manejo del directorio          */

/*=============================================================*/
get_nde(dir_pb)          /* Obtención de entrada siguiente del directorio */
/*=============================================================*/
/* Esta función devuelve un apuntador a la entrada siguiente del directorio.
   Abre el directorio si éste no estaba ya abierto.
   Si es necesario, se lee en el siguiente sector del directorio si se ha
   modificado el sector en curso, y es necesaria la escritura en el disco,
   ésta se realizará antes de la lectura del sector siguiente. */

/* Parámetros de entrada */
struct _dirpb *dir_pb;           /* Apuntador al bloque de parámetros del disco */

/* Parámetros de salida
   Devuelve un apuntador a la entrada siguiente del directorio en la
   memoria intermedia. Los indicadores de abierto y escritura de sector del
   directorio, en el bloque de parámetros, se inicializan si es necesario.
*/

{
if(!dir_pb -> dp_open)           /* Directorio no abierto */
        {
        if (!open_dir(dir_pb))   /* Inicialización y apertura del directorio */
                {
                err_dir(O_DIR,dir_pb);           /* Error en apertura */
                exit();
                }
                /* Fijación deliberada del apuntador a la entrada del directorio al final de la
                   memoria intermedia para forzar la lectura de un sector del directorio */
```

Figura 11-1. (Continuación.)

```
        dir_pb -> dp_entry = dir_pb -> dp_buffer + DIR_BSZ;
        dir_pb -> dp_write = 0;          /* Puesta a cero del indicador de escritura de sector */
        }

        /* Actualización del apuntador a la entrada del directorio a la entrada
           siguiente en la memoria intermedia. Comprobación de si el apuntador ha sobrepasado el
           final de la memoria intermedia y es necesaria la lectura de nuevo sector. */
if (++dir_pb -> dp_entry < dir_pb -> dp_buffer + DIR_BSZ)
        {
        return dir_pb -> dp_entry;       /* Devolución de apuntador a entrada siguiente */
        }

        /* Necesidad de desplazamiento a nuevo sector y lectura del mismo */

        /* No es necesaria la comprobación de final de directorioo o de
           desplazamiento al siguiente sector si el directorio ha sido abierto en
           este momento (el indicador de abierto no ha sido puesto a uno todavía) */
if (!dir_pb -> dp_open)
        dir_pb -> dp_open = 1;   /* Indicación de que el directorio se ha abierto ahora */
else
        {
        /* Comprobación de si el sector en curso en la memoria intermedia
           necesita ser escrito fuera del disco (por cambio) */
        if (dir_pb -> dp_write)
                {
                dir_pb -> dp_write = 0;          /* Puesta a cero de indicador */
                if(!rw_dir(W_DIR,dir_pb))        /* Escritura de sector de directorio */
                        {
                        err_dir(W_DIR,dir_pb);   /* Indicación de error de escritura */
                        exit();
                        }
                }

                /* Decrementación del número de entradas al directorio a procesar,
                   siempre cuatro entradas de 32 bytes por sector de 128 bytes */
        dir_pb -> dp_entrem -= 4;

                /* Fijación a uno del indicador final de directorio si número de entradas < 0 */
        if (dir_pb -> dp_entrem == 0)            /* Final de directorio */
                {
                dir_pb -> dp_end = 1;            /* Indicación de final */
                dir_pb -> dp_open = 0;           /* Indicación de directorio cerrado */
                return 0;                        /* Indicación de no más entradas */
                }

                /* Actualización de sector (y si es necesario pista y sector) */
        if (++dir_pb -> dp_sector == dir_pb -> dp_sptrk)
                {
                ++dir_pb -> dp_track;            /* Actualización de pista */
                dir_pb -> dp_sector = 0;         /* Puesta a cero de sector */
                }
        }

if(!rw_dir(R_DIR,dir_pb))        /* Lectura de nuevo sector de directorio */
        {
        err_dir(R_DIR,dir_pb);   /* Indicación de error en lectura */
        exit();
        }

        /* Puesta del apuntador a entrada del directorio a la primera entrada en mem. int. */
return dir_pb -> dp_entry = dir_pb -> dp_buffer;

} /* Final de get_nde */


/*****************************************************************/
open_dir(dir_pb)         /* Apertura del directorio */
/*****************************************************************/
/* Esta función "abre" el directorio de ficheros de un disco
   especificado para su procesamiento con las funciones
   rw_dir y next_dir. */

/* Parámetros de entrada */
struct _dirpb *dir_pb;   /* Apuntador al bloque de parámetros del directorio */
```

n

o

Figura 11-1. (Continuación.)

```
/* Parámetros de salida
   0 = Error, directorio no abierto
   1 = Directorio abierto para procesamiento
*/

{
struct _dpb *dpb;                    /* Bloque de parámetros de disco CP/M */

        /* Obtención de la dirección del bloque de parámetros de disco para el disco
           especificado en el bloque de parámetros de directorio */
if ((dpb = get_dpb(dir_pb -> dp_disk)) == 0)
        return 0;           /* Retorno indicando no existencia de DPB para este disco */

        /* Fijación de los campos restantes en el bloque de parámetros */
dir_pb -> dp_sptrk = dpb -> dpb_sptrk;   /* Sectores por pista */
dir_pb -> dp_track = dpb -> dpb_trkoff;  /* Desplazamiento de pistas del directorio */
dir_pb -> dp_sector = 0;                 /* Comienzo del directorio */
dir_pb -> dp_nument = dpb -> dpb_maxden+1; /* Número de entradas al directorio */
dir_pb -> dp_entrem = dir_pb -> dp_nument; /* Entradas a procesar */
dir_pb -> dp_end = 0;                    /* Indicación de no final */

        /* Fijación del número de bloques por entrada de directorio a 8 ó 16
           dependiendo del número de bloques */
dir_pb -> dp_nabpde = (dpb -> dpb_maxabn > 255 ? 8 : 16);
        /* Fijación del número de bloques (uno por encima del
           bloque mayor) */
dir_pb -> dp_nab = dpb -> dpb_maxabn;

        /* Fijación del tamaño de los bloques basado en el desplazamiento.
           Los valores posibles son 3 = 1k, 4 = 2k, 5 = 4k, 6 = 8k, 7 = 16k.
           con lo que un valor 16 es desplazado a la derecha (7 - bshift) bits. */
dir_pb -> dp_absize = 16 >> (7 - dpb -> dpb_bshift);

return 1;                /* Indicación de directorio abierto */

} /*  Final de open_dir  */
```

o

```
/*=============================================================*/
rw_dir(read_op,dir_pb)   /* Lectura/escritura de directorio */
/*=============================================================*/
/* Esta función lee/escribe el siguiente sector de 128 bytes del
   directorio abierto en curso. */

/* Parámetros de entrada */
short read_op;           /* Verdadero para lectura, falso (0) para escritura */
struct _dirpb *dir_pb;   /* Bloque de parámetros del directorio */

/* Parámetros de salida
   0 = Error operación no realizada
   1 = Operación finalizada
*/

{
struct _drb drb;                /* Disco requerido (por lectura/escritura del BIOS) */


drb.dr_disk = dir_pb -> dp_disk;         /* Fijación de petición de disco */
drb.dr_track = dir_pb -> dp_track;
drb.dr_sector = dir_pb -> dp_sector;
drb.dr_buffer = dir_pb -> dp_buffer;

if (read_op)
        {
        if (!rd_disk(&drb))      /* Lanzamiento de orden de lectura */
                return 0;        /* Indicación de error no datos disponibles */
        }
else
        {
        if (!wrt_disk(&drb))     /* Lanzamiento de orden de escritura */
                return 0;        /* Indicación de error no datos escritos */
        }
return 1;                        /* Indicación de operación finalizada */

} /* Final de rd_dir */
```

p

Figura 11-1. (Continuación.)

```
/*=================================================================*/
err_dir(opcode,dir_pb)          /* Visualización de errores de directorio */
/*=================================================================*/
/* Esta función visualiza un mensaje de error para dar cuenta de un error
   detectado por las funciones de administración del directorio open_dir y rw_dir. */
/* Parámetros de entrada */
short opcode;                   /* Intento de operación */
struct _dirpb *dir_pb;  /* Apuntador al bloque de parámetros del directorio */

{
printf("\n\007Error during ");

switch(opcode)
        {
        case R_DIR:
                printf("Reading");
                break;
        case W_DIR:
                printf("Writing");
                break;
        case O_DIR:
                printf("Opening");
                break;
        default:
                printf("Unknown Operation (%d) on",opcode);
        }

printf(" Directory on disk %c:. ",dir_pb -> dp_disk + 'A');

} /* Final de err_dir */


/*=================================================================*/
setscb(scb,fname,user,extent,length)    /* Fijación del bloque de control de búsqueda */
/*=================================================================*/
/* Esta función fija el bloque de control de búsqueda de acuerdo
   con el nombre de fichero especificado. El nombre de fichero
   puede presentarse en las formas siguientes:

        nombre-de-fichero
        nombre-de-fichero.tipo
        d:nombre-de-fichero.tipo
        *:nombre-de-fichero.tipo (refiriéndose a los discos)
        ABCD...NOP:nombre-de-fichero.tipo (refiriéndose a los discos especificados)

   La función fija el mapa de bits de acuerdo con los discos en los que se
   busca. Para cada uno de los discos seleccionados comprueba si se ha generado
   algún error en la selección del disco (es decir, si existen las tablas
   de disco en el BIOS para él). */

/* Parámetros de entrada */
struct _scb *scb;               /* Apuntador al bloque de control de búsqueda */
char *fname;                    /* Apuntador al nombre de fichero */
short user;                     /* Número de usuario a buscar */
short extent;                   /* Número de extensión a buscar */
int length;                     /* Número de bytes a comparar */

/* Parámetros de salida
   Ninguno.
*/

{
int disk;                       /* Número de disco comprobándose */
unsigned adisks;                /* Mapa de bits para los discos activos */

adisks = 0;                     /* Suposición de que no hay discos a buscar */

if (strscn(fname,":"))          /* Comprobación de ":" en nombre de fichero */
        {
        if (*fname == '*')      /* Comprobación de si "todos" los discos */
                {
                adisks = 0xFFFF;        /* Puesta a uno de todos los bits */
                }
        else                    /* Puesta a uno de los discos específicos */
                {
                while(*fname != ':')    /* Hasta encontrar ":" */
```

Figura 11-1. (Continuación.)

```
                        {
                        /* Construcción del mapa de bits por obtención del identificador
                           (A - P) de disco siguiente, convirtiéndolo en un número
                           de 0 a 15, desplazando un bit a la izquierda
                           cada vez y realizando la operación OR con los discos activos
                           en curso. */
                        adisks |= 1 << (toupper(*fname) - 'A');
                        ++fname;            /* Desplazamiento al carácter siguiente */
                        }
                ++fname;                    /* Ignorar ":" */
                }
        }
else    /* Utilizar únicamente disco en curso por defecto */
        {
                /* Fijación del bit correspondiente al disco en curso */
        adisks = 1 << bdos(GETDISK);
        }

setfcb(scb,fname);          /* Fijación del bloque de control de búsqueda como si fuera
                               un bloque de control de fichero. */

/* Realización de llamadas a la rutina SELDSK del BIOS
   para asegurar que todas las unidades de disco activas
   tienen tablas definidas en el BIOS. Si no existen puesta a cero
   del bit correspondiente del mapa de bits. */

for (disk = 0;              /* Comienzo con el disco A: */
     disk < 16;             /* Hasta el disco P: */
     disk++)                /* Utilización del disco siguiente */
        {
        if ( !((1 << disk) & adisks))
                continue;               /* Evitar la selección de discos no especificados */
        if (biosh(SELDSK,disk) == 0)    /* Realización de llamada a SELDSK del BIOS */
                {                       /* Devolución de 0 si disco no válido */
                /* Puesta a cero del bit correspondiente en la máscara,
                   realizando la operación AND con una máscara de bits
                   que tiene todos los demás bits a uno */
                adisks &= ((1 << disk) ^ 0xFFFF);
                }
        }

scb -> scb_adisks = adisks;     /* Puesta de mapa de bits en el SCB */
scb -> scb_userno = user;       /* Puesta del número de usuario */
scb -> scb_extent = extent;     /* Puesta del número de extensión */
scb -> scb_length = length;     /* Fijación del número de bytes a comparar */

} /* Final de setscb */

/*=================================================================*/
dm_clr(disk_map)                /* Borrado de mapa de disco (a cero) */
/*=================================================================*/
/* Esta función pone a cero todos los elementos del mapa de disco */

/* Parámetros de entrada */
unsigned disk_map[16][18];      /* Dirección de la matriz de enteros sin signo */

/* Parámetros de salida
   Ninguno.
*/

{
        /* ATENCION -- El 576 en la llamada a setmen que sigue se basa en que
           la matriz del mapa de disco es de 16x18 -- es decir, 288 enteros
           sin signo; por tanto, 576 bytes. */
setmem(disk_map,576,'\0');      /* Llenado del array con ceros */

} /* Final de dm_clr */

/*=================================================================*/
dm_disp(disk_map,adisks)        /* Visualización del mapa de disco */
/*=================================================================*/
/* Esta función visualiza los elementos del mapa de disco mostrando
   la cuenta en cada elemento. Un valor cero en un elemento
   se muestra como blancos. Por ejemplo:
```

Figura 11-1. (Continuación.)

```
     0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  Utilizados libres
A: 123      20  98          202     199 101 211                         954   70

   Se imprimirán únicamente las líneas para los discos activos (como indica
   el mapa de bits). */

/* Parámetros de entrada */
unsigned disk_map[16][18];          /* Apuntador a matriz del mapa de disco */
unsigned adisks;                    /* Mapa de bits de discos activos */

{
#define USED_COUNT 16               /* Número de "usuario" para entidades utilizadas */
#define FREE_COUNT 17               /* Número de "usuario" para entidades libres */

int disk;                           /* Número de disco en curso */
int userno;                         /* Número de usuario en curso */
unsigned dsum;                      /* Suma de entradas para un disco dado */

printf("\n      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15 Utilizados libres");

for (disk = 0;                      /* Comienzo con el disco A: */
     disk < 16;                     /* Hasta el disco P: */
     disk++)                        /* Disco siguiente */
        {
        if (!(adisks & (1 << disk)))    /* Comprobación de disco activo */
                continue;               /* No, lo ignora */

        printf("\n%c: ",disk + 'A');    /* Visualización de número de disco */

        dsum = 0;                       /* Puesta a cero de la suma para este disco */
        for (userno = 0;                /* Comienzo con usuario 0 */
             userno < 16;               /* Hasta usuario 15 */
             userno++)                  /* Siguiente número de usuario */
                {
                dsum += disk_map[disk][userno]; /* Realización de la suma */
                }

        if (dsum)           /* Comprobación de salida para este disco,
                               si no visualizar d: Ninguno */
                {
                /* Imprimir número o blancos */
                for (userno = 0;            /* Comienzo por usuario 0 */
                     userno < 16;           /* Hasta usuario 15 */
                     userno++)              /* Siguiente número de usuario */
                        {
                        if (disk_map[disk][userno])
                                printf("%4d",disk_map[disk][userno]);
                        else
                                printf("    ");
                        }
                }
        else                /* No hay salida para este disco */
                {
                printf("  -- None --
                }
        printf("  %4d %4d",disk_map[disk][USED_COUNT],disk_map[disk][FREE_COUNT]);
        }

} /* Final de dm_disp */

/*****************************************************************/
get_dpb(disk)               /* Obtención de dirección del bloque de parámetros de disco */
/*****************************************************************/
/* Esta función devuelve la dirección del bloque de parámetros
   del disco (localizada en el BIOS). */

/* Parámetros de entrada */
char disk;                  /* Disco lógico que precisa la dirección del DPB */

/* Parámetros de salida
       0 = Disco lógico no válido
       NZ = Apuntador al bloque de parámetros de disco
*/

{
if (biosh(SELDSK,disk) == 0)            /* Realización de llamada a SELDSK del BIOS */
        return 0;                       /* Disco no válido */
```

Figura 11-1. (Continuación.)

```
    bdos(SETDISK,disk);                    /* Utilización de la función SETDISK del BDOS */
    return bdos(GETDPARM);                 /* Obtención del bloque de parámetros de disco */
    } /*  Final de get_dpb  */                                                              u


    /*     Funciones de tabla de códigos       */

    /* Muchos programas que interaccionan con el usuario deben
       aceptar parámetros de éste como nombres y traducir este nombre
       a algún código interno.
       Es posible también el proceso inverso, examinando
       el valor de una variable y determinando a qué nombre ASCII
       ha sido puesta.

       Un ejemplo es la fijación de velocidad de transmisión (baud rate).
       El usuario desea entrar "19200" y que una vez traducido
       sea enviado al chip correspondiente. Viceversa la variable
       de velocidad de transmisión fijada con anterioridad
       debe poderse examinar y generar la cadena "19200" para que el
       usuario puede visualizar este valor.

       La tabla de códigos se utiliza para facilitar esta tarea.
       Cada elemento de la tabla consta lógicamente de:
            Un código (entero sin signo)
            Una cadena de caracteres ASCII (realmente un apuntador a ella) */          v

    /*====================================================================*/
    ct_init(entry,code,string)          /* Inicialización tabla de códigos */
    /*====================================================================*/
    /* Esta función inicializa una entrada específica en la tabla de códigos
       con un código y un apuntador a cadena.

");     NOTA: Por convenio, la última entrada a una tabla de códigos
       determinada tiene el código CT_SNF (cadena no encontrada). */

    /* Parámetros de entrada */
    struct _ct *entry;                  /* Apuntador a entrada de tabla de códigos */
    int code;                           /* Código a almacenar en dicha entrada */
    char *string;                       /* Apuntador a cadena para esta entrada */

    /* Parámetros de salida
       Ninguno.
    */

    {
    entry -> _ct_code = code;                   /* Puesta de código_ct */
    entry -> _ct_sp = string;                   /* Puesta de apuntador a cadena */
    } /*  Final de ct_inti  */

    /*====================================================================*/
    unsigned
    ct_parc(table,string)               /* Parámetros - código de retorno */
    /*====================================================================*/
    /* Esta función examina en la tabla especificada la coincidencia con
       una cadena y retorna el código que le corresponde.
       Si encuentra únicamente una coincidencia en la tabla, devuelve
       el código correspondiente. Si no encuentra coincidencia
       o más de una, devuelve un código de error CT_SNF (cadena no encontrada).
       Esta función se ha diseñado específicamente para el procesamiento
       de los parámetros de orden.
       Nótese que la comparación se realiza después de la conversión amayúsculas
       (es decir, "CADENA" coincide con "cadena"). La comparación de subcadenas se utiliza
       para que sea necesario entrar el mínimo número de características para obtener
       una respuesta no ambigua. Por ejemplo, si la tabla contiene:                    w

                  Código    Valor
                  1         "MANZANAS"
                  2         "NARANJAS"
                  3         "MELOCOTONES"

       En respuesta a "N" devolverá el código = 2, pero "M" es ambiguo.
       Es necesario "MA" o "ME". */

    struct _ct *table;                  /* Apuntador a tabla */
    char *string;                       /* Apuntador a cadena clave */
```

Figura 11-1. (Continuación.)

```
{
int mcode;                          /* Código de coincidencia a devolver */
int mcount;                         /* Cuenta del número de coincidencia */

mcode = CT_SNF;                     /* Suposición de error */
mcount = 0;                         /* Puesta a cero de cuenta de coincidencias */

while(table -> _ct_code != CT_SNF) /* No final de tabla */
        {
        /* Comparación de la respuesta en teclado con entrada de tabla
           utilizando la comparación de subcadenas. */
        if (usstrcmp(table -> _ct_sp,string) == 0)
                {
                mcount++;           /* Actualización cuenta de coincidencias */
                mcode = table -> _ct_code;        /* Almacenar código */
                }
        table++;                    /* Desplazamiento a entrada siguiente */
        }

if (mcount == 1)                    /* Sólo una coincidencia */
        return mcode;               /* Devolución de código de coincidencia */
else                                /* Ilegal o ambiguo */
        return CT_SNF;

} /* Final de ct_parc */
```

w

```
/*================================================================*/
unsigned
ct_code(table,string)    /* Devolución de código para cadena */
/*================================================================*/
/* Esta función examina la tabla especificada
   para la cadena especificada. En caso
   contrario devuelve CT_SNF (cadena no
   encontrada).
   Al contrario que la anterior compara cada carácter de la cadena de entrada
   y si hay coincidencia devuelve el código de la primera coincidencia encontrada. */

/* Parámetros de entrada */
struct _ct *table;        /* Apuntador a tabla */
char *string;             /* Apuntador a cadena */

/* Parámetros de salida
   Código -- si cadena encontrada
   CT_SNF -- si cadena no encontrada
*/

{
while(table -> _ct_code != CT_SNF)        /* Para todas las entradas en la tabla */
        {
        if (ustrcmp(table -> _ct_sp,string) == 0) /* Comparación de cadenas */
                return table -> _ct_code;        /* Devolución de código */
        table++;                          /* Desplazamiento a entrada siguiente */
        }
return CT_SNF;                            /* Cadena no encontrada */

} /* Final de ct_code */
```

x

```
/*================================================================*/
ct_disps(table) /* Visualización de todas las cadenas de una tabla especifica */
/*================================================================*/
/* Esta función visualiza todas las cadenas de una tabla específica.
   Se utiliza para indicar respuestas válidas a entrada de operador. */

/* Parámetros de entrada */
struct _ct *table;                /* Apuntador a tabla */

/* Parámetros de entrada
   Ninguno.
*/

{
while(table -> _ct_code != CT_SNF)        /* No final de tabla */
        {
        printf("\n\t\t%s",table -> _ct_sp);      /* Impresión de cadena */
        table++;                          /* Desplazamiento a entrada siguiente */
        }
```

y

Figura 11-1. (Continuación.)

```
putchar('\n');                          /* Añadir retorno al final */

} /* Final de ct_disps */

/*==================================================================*/
ct_index(table,string) /* Devolución de índice para una cadena dada */
/*==================================================================*/
/* Esta función examina la tabla especificada y devuelve
   el INDICE de la entrada conteniendo la cadena coincidente.
   Se utilizan todos los caracteres en la comparación después
   de convertirlos a mayúsculas. */

/* Parámetros de entrada */
struct _ct *table;                  /* Apuntador a tabla */
char *string;                       /* Apuntador a cadena */

/* Parámetros de salida
   Indice de la cadena coincidente, o
   CT_SNF si no hay coincidencia.
*/

{
int index;                          /* Valor del índice en curso */

index = 0;                          /* Inicialización del índice */

while(table -> _ct_code != CT_SNF)       /* No al final de la tabla */

        {
        if (ustrcmp(table -> _ct_sp,string) == 0)
                return index;       /* Devolución de índice */
        table++;                    /* Desplazamiento a entrada siguiente */
        index++;                    /* Actualización del índice */
        }
return CT_SNF;              /* Cadena no encontrada */

} /* Final de ct_index */

/*==================================================================*/
char *ct_stri(table,index)          /* Obtención de cadena de acuerdo con el índice */
/*==================================================================*/
/* Esta función devuelve un apuntador a la cadena en la entrada
   de tabla especificada por el índice. */

/* Parámetros de entrada */
struct _ct *table;                  /* Apuntador a tabla */
int index;                          /* Indice en la tabla */

{
struct _ct *entry;                  /* Apuntador a entrada */
        entry = table[index];       /* Apunta a una entrada */
        return entry -> _ct_sp;     /* Devolución de apuntador a cadena */

} /* Final de ct_stri */

/*==================================================================*/
char *ct_strc(table,code)  /* Obtención de cadena de acuerdo con un código */
/*==================================================================*/
/* Esta función examina la tabla especificada y devuelve
   al apuntador a cadena de caracteres en la entrada con código
   coincidente o un apuntador a la cadena "desconocido" si no
   encuentra el código. */

/* Parámetros de entrada */
struct _ct *table;                  /* Apuntador a tabla */
unsigned  code;                     /* Código */

{
while(table -> _ct_code != CT_SNF)       /* Hasta final de tabla */
        {
        if (table -> _ct_code == code)  /* Comprobación de coincidencia de código */
                return table -> _ct_sp; /* Sí, devolución de apuntador a cadena */
        table++;                        /* No, desplazamiento a entrada siguiente */
```

y

z

aa

bb

Figura 11-1. (Continuación.)

```
        }
return "Unknown";
}


/*      Funciones de vectores de bits    */                                    bb

/* Estas funciones manipulan vectores de bits. Un vector de bits
   es un grupo de bits adyacentes en paquetes de ocho por byte. Cada vector
   de bits tiene la estructura definida en el fichero LIBRARY.H.

   Los vectores de bits se utilizan principalmente para manipular
   los vectores de asignación del sistema operativo y otros valores,
   cuya mejor representación es una serie de bits. */


/*============================================================*/
bv_make(bv,bytes)          /* Creación de un vector de bits y puesta a ceros */
/*============================================================*/
/* Esta función utiliza la asignación de memoria interna de C,
   alloc, para asignar el espacio de memoria necesario y poner
   a cero el vector. */

/* Parámetros de entrada */
struct _bv *bv;            /* Apuntador a vector de bits */
unsigned bytes;            /* Número de bytes del vector de bits */

/* Parámetros de salida
   NZ = Vector creado
   0 = No suficiente memoria para crear vector                               cc
*/

{
if(!(bv -> bv_bits = alloc(bytes)))        /* Petición de memoria */
        return 0;                          /* Petición fallida */

bv -> bv_bytes = bytes;                    /* Fijación de longitud */
bv -> bv_end = bv -> bv_bits + bytes;      /* Fijación de apuntador al final */

bv_fill(bv,0);                             /* Rellenar vector con ceros */
return 1;

} /* Final de bv_make */


/*============================================================*/
bv_fill(bv,value)          /* Puesta de valor en el vector de bits */
/*============================================================*/
/* Esta función pone un valor especificado en el vector de bits
   especificado.
   Existe únicamente por razones de consistencia y para aislar
   el cuerpo principal del programa de funciones estándar como
   setmen. */

/* Parámetros de entrada */
struct _bv *bv;            /* Apuntador a vector de bits */                  dd
char value;                /* Valor a situar en él */

/* Parámetros de salida
   Ninguno.
*/


{
/*      dirección       longitud        valor */
setmem(bv -> bv_bits,bv -> bv_bytes,value);
}

/*============================================================*/
bv_set(bv,bitnum)                  /* Puesta a uno del bit especificado */
/*============================================================*/
/* Esta función pone a uno el bit especificado del vector
   de bits. */                                                               ee

/* Parámetros de entrada */
struct _bv *bv;                    /* Apuntador al vector de bits */
unsigned bitnum;                   /* Número de bit a fijar */
```

Figura 11-1. (Continuación.)

```
/* Parámetros de salida
   Ninguno.
*/

{
unsigned byte_offset;           /* Bytes de desplazamiento en el vector de bits */

if ((byte_offset = bitnum >> 3) > bv -> bv_bytes)
        return 0;               /* El número de bits sobrepasa el final del vector */

/* Puesta a uno del bit adecuado del vector. El byte de desplazamiento
   se ha calculado. El número de bit en el byte se calcula realizando
   la operación AND del número de bits con 0x07.
   El bit especificado se opera entonces (OR) con el vector */

bv -> bv_bits[byte_offset] |= (1 << (bitnum & 0x7));

return 1;                       /* Indicación de la operación realizada */

/* Final de bv_set */
```

ee

```
/*=============================================================*/
bv_test(bv,bitnum)              /* Test del número de bit especificado */
/*=============================================================*/
/* Esta función devuelve un valor que refleja el valor actual
   del bit especificado. */

/* Parámetros de entrada */
struct _bv *bv;                 /* Apuntador al vector de bits */
unsigned bitnum;                /* Número de bit a examinar */

/* Parámetros de salida
   Ninguno.
*/

{
unsigned byte_offset;           /* Bytes de desplazamiento en el vector de bits */

if ((byte_offset = bitnum >> 3) > bv -> bv_bytes)
        return 0;               /* El número de bit sobrepasa el final del vector */

/* Puesta a uno del bit adecuado del vector. El byte de desplazamiento
   se ha calculado. El número de bit en el byte se calcula realizando
   la operación AND del número de bit con 0x07.
   El bit especificado se opera entonces (OR) con el vector. */

return bv -> bv_bits[byte_offset] & (1 << (bitnum & 0x7));

} /* Final de bv_tests */
```

ff

```
/*=============================================================*/
bv_nz(bv)                       /* Test de vector de bits distinto de cero */
/*=============================================================*/
/* Esta función comprueba cada uno de los bytes
   del vector especificado y vuelve indicando si existe algún bit a uno
   en el vector. */

/* Parámetros de entrada */
struct _bv *bv;                 /* Apuntador al vector */

/* Parámetros de salida */
   NZ = Uno o más bits del vector a uno
   0 = Todos los bits están a 0
*/

{
char *bits;                     /* Apuntador a los bits del vector */

bits = bv -> bv_bits;           /* Fijación de apuntador de trabajo */

while (bits != bv -> bv_end)    /* Para vector de bits completo */
        {
        if (*bits++)            /* Si distinto de cero */
                return bits--;  /* Devolución de apuntador al byte NZ */
```

gg

Figura 11-1. (Continuación.)

```
        }
return 0;                          /* Indicación de vector cero */

} /* Final by_nz */
```

gg

```
/*==================================================================*/
bv_and(bv3,bv1,bv2)                /* bv3 = bv1 & bv2 */
/*==================================================================*/
/* Esta función realiza la operación AND entre los bytes de los vectores
   de bits 1 y 2 almacenando el resultado en el vector 3. */

/* Parámetros de entrada */
struct _bv *bv1;                   /* Apuntador a vector de bits de entrada */
struct _bv *bv2;                   /* Apuntador a vector de bits de entrada */

/* Parámetros de salida */
struct _bv *bv3;                   /* Apuntador a vector de bits de salida */

{
char *bits1, *bits2, *bits3;       /* Apuntadores de trabajo a los vectores de bits */

bits1 = bv1 -> bv_bits;            /* Inicialización de apuntadores de trabajo */
bits2 = bv2 -> bv_bits;
bits3 = bv3 -> bv_bits;

        /* Realización de la operación AND hasta el final de alguno
           de los vectores de bits */
while (bits1 != bv1 -> bv_end &&
       bits2 != bv2 -> bv_end &&
       bits3 != bv3 -> bv_end)
        {
                *bits3++ = *bits1++ & *bits2++; /* bv3 = bv1 & bv2 */
        }
} /* Final de bv_and */
```

hh

```
/*==================================================================*/
bv_or(bv3,bv1,bv2)                 /* bv3 = bv1 or bv2 */
/*==================================================================*/
/* Esta función realiza la operación OR entre los bytes de los vectores
   1 y 2 almacenando el resultado en el vector 3. */

/* Parámetros de entrada */
struct _bv *bv1;                   /* Apuntador a vector de bits de entrada */
struct _bv *bv2;                   /* Apuntador a vector de bits de entrada */

/* Parámetros de salida */
struct _bv *bv3;                   /* Apuntador a vector de bits de salida */

{
char *bits1, *bits2, *bits3;       /* Apuntadores de trabajo a los vectores de bits */

bits1 = bv1 -> bv_bits;            /* Inicialización de apuntadores de trabajo */
bits2 = bv2 -> bv_bits;
bits3 = bv3 -> bv_bits;

        /* Realización de la operación OR hasta el final de alguno
           de los vectores de bits. */
while (bits1 != bv1 -> bv_end &&
       bits2 != bv2 -> bv_end &&
       bits3 != bv3 -> bv_end)
        {
                *bits3++ = *bits1++ | *bits2++; /* bv3 = bv1 or bv2 */
        }
} /* Final de bv_or */
```

ii

```
/*==================================================================*/
bv_disp(title,bv)                  /* Visualización de vectores de bits */
/*==================================================================*/
/* Esta función visualiza en hexadecimal el contenido del vector de bits
   especificado. Se utiliza normalmente para depuración de programas. */

/* Parámetros de entrada */
char *title;                       /* Título para la visualización */
struct _bv *bv;                    /* Apuntador al vector de bits */
```

jj

Figura 11-1. (Continuación.)

```
 /* Parámetros de salida
    Ninguno.
*/

{
char *bits;                          /* Apuntador de trabajo */
unsigned byte_count;                 /* Contador utilizado para formatear salida */
unsigned bit_count;                  /* Contador para procesamiento de los bits de un byte */
char byte_value;                             /* Valor a visualizar */

printf("\nBit Vector : %s",title);            /* Visualización de título */

bits = bv -> bv_bits;                        /* Fijación de apuntador de trabajo */
byte_count = 0;                              /* Inicialización del contador */

while (bits != bv -> bv_end)                 /* Para vector completo */
        {
        if (byte_count % 5 == 0)             /* Comprobación de nueva línea */
                                             /* Visualización número de bit */
                printf("\n%4d : ",byte_count << 3);

        byte_value = *bits++;        /* Obtención del byte siguiente del vector */

        for (bit_count = 0; bit_count < 8; bit_count++)
                {
                /* Visualización del bit más a la izquierda, a continuación desplazamiento
                   a la izquierda un bit */
                if (bit_count == 4) putchar(' '); /* Separador
                putchar((byte_value & 0x80) ? '1' : '0');
                byte_value <<= 1;            /* Desplazamiento a la izquierda */
                }
        printf("  ");                                /* Separador */

        byte_count++;    /* Actualización contador de bytes */
        }
} /* Final de bv_disp */

/* Final de LIBRARY.C */
```

jj

Figura 11-1. (Continuación.)

Asociada con la biblioteca de funciones hay otra parte de programa fuente denominada "LIBRARY.H", que se muestra en la figura 11-2. Este fichero "de cabeza" debe ser incluido al principio de cada programa que llama a una función cualquiera de la biblioteca.

Por razones de claridad, este capítulo describe las funciones más sencillas en primer lugar, seguidas por las más complicadas y finalmente por los programas de utilidad que utilizan las funciones.

Algunas funciones de la biblioteca y algunas definiciones de la cabeza de biblioteca no son utilizadas por las utilidades mostradas en este capítulo. Se han incluido para ilustrar técnicas y porque pueden ser útiles para otros programas de utilidad que se pueden escribir.

```
#define LIBVN "1.0"      /* Número de versión de biblioteca */

/* Este fichero contiene grupos de definiciones útiles.
   Debe incluirse al principio de cualquier programa
   que utilice funciones de LIBRARY.C */


/* Definición para realizar pequeñas modificaciones al lenguaje C. */
#define short char               /* Las abreviaciones no están soportadas directamente */
```

a

Figura 11-2. LIBRARY.H, programa que debe ser incluido al principio de cualquier programa que llame a funciones LIBRARY en la figura 11-1.

```
/* Una de las funciones de biblioteca utiliza la función C del BDS,
   alloc, para asignación de memoria. Las definiciones siguientes
   son para alloc. */

struct _header                  /* Cabecera para bloque de memoria asignando */
        {
        struct _header *_ptr;   /* Apuntador a cabecera siguiente en la serie */
        unsigned _size;         /* Número de bytes del bloque asignado */
        };
struct _header _base;           /* Declaración de la primera cabecera de la serie */
struct _header *_allocp;        /* Utilizado por funciones alloc() y free() */

/* Números de llamada a la función del BDOS */

#define SETDISK 14      /* Selección de disco */
#define SEARCHF 17      /* Búsqueda del primer caso */
#define SEARCHN 18      /* Búsqueda del siguiente */
#define DELETEF 19      /* Borrado del fichero */
#define GETDISK 25      /* Obtención de disco por defecto */
#define SETDMA  26      /* Fijación de dirección de DMA (lectura/escritura) */
#define GETDPARM 31     /* Obtención dirección de bloque de parámetros del disco */
#define GETUSER 32      /* Obtención número de usuario en curso */
#define SETUSER 32      /* Fijación número de usuario en curso */

/* Llamadas directas al BIOS
   Estas definiciones son para llamadas directas al BIOS.
   ATENCION:   Su utilización hace menos transportable el programa.
   Cada símbolo se relaciona con el salto correspondiente en el
   vector de saltos del BIOS.
   Sólo están definidas las entradas más usuales. */

#define CONST   2       /* Estado de consola */
#define CONIN   3       /* Entrada de consola */
#define CONOUT  4       /* Salida de consola */
#define LIST    5       /* Salida de impresión (listado) */
#define AUXOUT  6       /* Salida auxiliar */
#define AUXIN   7       /* Entrada auxiliar */

#define HOME    8       /* Puesta a cero del disco (pista y sector) */
#define SELDSK  9       /* Selección de disco lógico */
#define SETTRK  10      /* Fijación de pista */
#define SETSEC  11      /* Fijación de sector */
#define SETDMA  12      /* Fijación de dirección de DMA */
#define DREAD   13      /* Lectura de disco */
#define DWRITE  14      /* Escritura en disco */
#define LISTST  15      /* Estado de listado */
#define SECTRN  16      /* traducción de sector */
#define AUXIST  17      /* Estado de entrada auxiliar */
#define AUXOST  18      /* Estado de salida auxiliar */

                        /* Entradas "privadas" al vector de saltos */
#define CIOINIT 19      /* Inicialización de E/S de caracteres específicos */
#define SETDOG  20      /* Activación temporizador-centinela */
#define CBGADDR 21      /* Obtención de dirección del bloque de configuración */

/* Definiciones para acceder al bloque de configuración */

#define CB_GET 21               /* Número de salto del BIOS a rutina de acceso */
#define DEV_INIT 19             /* Salto del BIOS para inicialización de dispositivo */

#define CB_DATE 0               /* Fecha en ASCII */
#define CB_TIMEA 1              /* Hora en ASCII */
#define CB_DTFLAGS 2            /* Indicadores de fecha y hora */
#define TIME_SET 0x01           /* Este bit de NZ indica que se ha fijado la fecha */
#define DATE_SET 0x02           /* Este bit de NZ indica que se ha fijado la hora */

#define CB_FIP 3                /* Apuntador a entrada forzada */
#define CB_SUM 4                /* Mensaje de inicialización del sistema */

#define CB_CI 5                 /* Entrada de consola */
#define CB_CO 6                 /* Salida de consola */
#define CB_AI 7                 /* Entrada auxiliar */
#define CB_AO 8                 /* Salida auxiliar */
```

b

c

d

e

Figura 11-2. (Continuación.)

```
#define CB_LI 9                 /* Entrada de listado */
#define CB_LO 10                /* Salida de impresión */

#define CB_DTA 11               /* Direcciones de tabla de dispositivo */
#define CB_C1224 12             /* Indicador de formato de reloj de 12/24 horas */
#define CB_RTCTR 13             /* Número de pulsos por segundo del reloj de tiempo real */

#define CB_WDC 14               /* Contador-centinela */
#define CB_WDA 15               /* Dirección de centinela */

#define CB_FKT 16               /* Tabla de teclas función */
#define CB_COET 17              /* Tabla de escapes de salida consola */

#define CB_DO_IS 18             /* Flujo de inicialización de dispositivo 0 */
#define CB_DO_BRC 19            /* Constante de velocidad de transmisión de dispositivo 0 */

#define CB_D1_IS 20             /* Flujo de inicialización de dispositivo 1 */
#define CB_D1_BRC 21            /* Constante de velocidad de transmisión de dispositivo 1 */

#define CB_D2_IS 22             /* Flujo de inicialización de dispositivo 2 */
#define CB_D2_BRC 23            /* Constante de velocidad de transmisión de dispositivo 2 */

#define CB_IV 24                /* Vector de interrupciones */
#define CB_LTCBO 25             /* Desplazamiento del bloque de configuración a largo plazo */
#define CB_LTCBL 26             /* Longitud del bloque de configuración a largo plazo */

#define CB_PUBF 27              /* Indicador de ficheros públicos */
#define CB_MCBUF 28             /* Memoria intermedia multiorden */
#define CB_POLLC 29             /* Indicador de análisis de consola */

        /* Números de dispositivo y nombres para dispositivos físicos */
        /* NOTA: Cámbiense estas definiciones para computadora particular */

#define T_DEVN 0                /* terminal */
#define M_DEVN 1                /* Modem */
#define P_DEVN 2                /* Impresora */

#define MAXPDEV 2               /* Número máximo de dispositivo físico */

        /* Nombres para dispositivos físicos */

#define PN_T "TERMINAL"
#define PN_M "MODEM"
#define PN_P "PRINTER"

        /* Estructura y definiciones para teclas-función */

#define FK_ILENGTH 2            /* Número de caracteres de entrada cuando se pulsa tecla-función
                                   NOTA: No debe incluirse el escape. */
#define FK_LENGTH 16            /* Longitud de la cadena sin incluir fk_term */
#define FK_ENTRIES 18           /* Número de entradas de tecla-función en la tabla */

struct _fkt                     /* Tabla de teclas-función */
        {
        char fk_input[FK_ILENGTH];      /* Primer carácter no en la tabla */
        char fk_output[FK_LENGTH];      /* Salida de cadena de caracteres */
        char fk_term;                   /* Carácter de terminación */
        };


/* Definiciones y estructura para tablas de dispositivo */

        /* Bits de protocolo */
        /* Nota: Si el bit más significativo
           está a uno, entonces la función set_proto
           realizará el OR con el valor. Ello
           permite que Input DTR coexista con el
           protocolo XON o ETX. */

#define DT_ODTR 0x8004          /* Salida DTR alto para enviar (OR-ed in) */
#define DT_OXON 0x0008          /* Salida de XON */
#define DT_OETX 0x0010          /* Salida de ETX/ACK */

#define DT_IRTS 0x8040          /* Entrada RTS (OR-ed in) */
#define DT_IXON 0x0080          /* Entrada XON */
```

e

f

g

h

i

Figura 11-2. (Continuación.)

```
@define ALLPROTO 0xDC              /* Combinados todos los protocolos */

struct _dt                         /* Tabla de dispositivo */
        {
        char dt_f1[14];            /* Rellenador */
        char dt_st1;               /* Byte 1 de estado -- tiene indicador de protocolo */
        char dt_st2;               /* Byte de estado 2 */
        unsigned dt_f2;            /* Rellenador */
        unsigned dt_etxml;         /* Longitud de mensaje ETX/ACK */
        char dt_f3[12];            /* Rellenador */
        } ;

/* Valores devueltos por comp_fname (comparación de nombre de fichero ) */

@define NAME_EQ 0          /* Nombres iguales */
@define NAME_LT 1          /* Nombre menor igual que máscara */
@define NAME_GT 2          /* Nombre mayor igual que máscara */
@define NAME_NE 3          /* Nombre distinto (y comparación ambigua) */

/* Estructura del bloque de control de ficheros CP/M estándar */

@define FCBSIZE 36                 /* Definición de la longitud total del FCB */

struct _fcb
        {
        short fcb_disk;            /* Disco lógico (0 = defecto) */
        char fcb_fname[11];        /* Nombre-de-fichero.tipo (con atributos) */
        short fcb_extent;          /* Extensión en curso */
        unsigned fcb_s12;          /* Reservado para CP/M */
        short fcb_reccnt;          /* Cuenta de registros utilizados en extensión en curso */
        union                      /* Los bloques pueden ser */
                {                  /* De un solo byte o de dos bytes */
                short fcbab_short[16];
                unsigned fcbab_long[8];
                } _fcbab;
        short fcb_currec;          /* Registro en curso en la extensión */
        char fcb_ranrec[3];        /* Registro para lectura/escritura aleatoria */
        };

/* Bloque de parámetros utilizados para llamadas a rutinas de manejo de directorio */

@define DIR_BSZ 128                /* Tamaño de la memoria intermedia de directorio */

struct _dirpb
        {
        short dp_open;             /* 0 para petición de apertura de directorio */
        short dp_end;              /* NZ cuando se ha llegado al final del directorio */
        short dp_write;            /* NZ para escribir el sector en curso en el disco */
        struct _dir *dp_entry;     /* Apuntador a la entrada de directorio en la memoria intermedia */
        char dp_buffer [DIR_BSZ          /* Memoria intermedia de sector del directorio */
        char dp_disk;              /* Disco lógico en curso */
        int dp_track;              /* Pista de comienzo */
        int dp_sector;             /* Sector de comienzo */
        int dp_nument;             /* Número de entradas al directorio */
        int dp_entrem;             /* Entradas en fase de procesamiento */
        int dp_sptrk;              /* Número de sectores por pista */
        int dp_nabpde;             /* Número de bloques por entrada al directorio */
        unsigned dp_nab;           /* Número de bloques */
        int dp_absize;             /* Tamaño de los bloques (en Kbytes) */
        };

/* La función err_dir se utiliza para tomar en cuenta los errores
   encontrados por las rutinas de manejo del directorio open_dir y rw_dir.
   Err_dir necesita de un parámetro para definir la operación
   realizada cuando ha ocurrido el error. Las definiciones siguientes
   representan las operaciones posibles. */

@define W_DIR   0          /* Escritura del directorio */
@define R_DIR   1          /* Lectura del directorio */
@define O_DIR   2          /* Apertura del directorio */
```

i

j

k

l

m

Figura 11-2. (Continuación.)

```
/* Bloque de parámetros del disco mantenidos por CP/M */

struct _dpb
        {
        unsigned dpb_sptrk;         /* Sector por pista */
        short dpb_bshift;           /* Desplazamiento de bloques */
        short dpb_bmask;            /* Máscara de bloques */
        short dpb_emask;            /* Máscara de extensión */
        unsigned dpb_maxabn;        /* Número máximo de bloque */
        unsigned dpb_maxden;        /* Número máximo de entrada al directorio */
        short dpb_rab0;             /* Bloques reservados para */
        short dpb_rab1;             /*  bloques del directorio */
        unsigned dpb_diskca;        /* Area de trabajo de cambio de disco */
        unsigned dpb_trkoff;        /* Desplazamiento de pista */
        };                                                              n


/* Formato de entrada del directorio de disco */

struct _dir {
        char de_userno;             /* Número de usuario o 0xE5 si entrada libre */
        char de_fname[11];          /* Nombre de fichero [8] y tipo [3] */
        int de_extent;              /* Número de extensión de esta entrada */
        int de_reccnt;              /* Número de registros de 128 bytes utilizados
                                       en el último bloque */
        union                       /* Los bloques pueden ser de un único byte
                {                   /*  o de dos bytes */
                short de_short[16];
                unsigned de_long[8];
                } _dirab;
        };                                                              o


/* Parámetros requeridos por el disco para lectura/escritura a nivel del BIOS */

struct _drb
        {
        short dr_disk;              /* Disco lógico A = 0, B = 1... */
        unsigned dr_track;          /* Pista (para SETTRK) */
        unsigned dr_sector;         /* Sector (para SETSEC) */
        char *dr_buffer;            /* Dirección de la memoria intermedia (para SETDMA) */
        } ;                                                             p


/* Bloque de control de búsqueda utilizado por las funciones de rastreo de directorio */

struct _scb
        {
        short scb_userno;           /* Número(s) de usuario a contrastar */
        char scb_fname[11];         /* Nombre y tipo de fichero */
        short scb_extent;           /* Número de extensión */
        char unused[19];            /* Bytes ficticios para hacer esto como
                                       un bloque de control de fichero */
        short scb_length;           /* Número de bytes a comparar */
        short scb_disk;             /* Disco en curso a buscar */
        unsigned scb_adisks;        /* Mapa de bits de los discos a buscar
                                       el bit más a la derecha es para el disco A:: */
        } ;                                                             q


/* Definiciones relacionadas con la tabla de códigos */

#define CT_SNF 0xFFFF    /* Cadena no encontrada */

struct _ct               /* Definición de la estructura de la tabla de códigos */
        {
        unsigned _ct_code;          /* Código */
        char *_ct_sp;               /* Apuntador a cadena */
        };                                                              r
```

Figura 11-2. (Continuación.)

```
/* Estructura para los vectores de bits */

struct _bv
        {
        unsigned bv_bytes;      /* Número de bytes del vector */
        char *bv_bits;          /* Apuntador al primer byte del vector de bits */
        char *bv_end;           /* Apuntador al byte que sigue al vector de bits */
        } ;


/* Final de LIBRARY.H */
```

Figura 11-2. (Continuación.)

Funciones de biblioteca

Esta sección describe las funciones de biblioteca y las secciones desde el fichero de cabeza que deben ser incluidas al principio de cada programa de utilidad.

Un cambio menor a lenguaje C

Un problema menor con el compilador C BDS es que no soporta enteros "cortos", o números enteros que tengan sólo un único byte de longitud. Es conveniente declarar ciertos valores como cortos para que sirvan de recordatorio a la definición de tipo estándar. Por consiguiente, el compilador C BDS deberá ser "engañado" declarando estos valores como si fueran simples caracteres. Para ello, el fichero de cabeza de biblioteca contiene la declaración

```
#define short char.
```

que se muestra en la figura 11-2, sección a.

El "#define" indica a la primera parte del compilador C, el preprocesador, que sustituya la hilera "char" (que declara en una variable tipo carácter) siempre que encuentre la cadena "short" (que normalmente declararía un número entero corto en C estándar).

Obsérvese que las cadenas de caracteres incluidas entre "/*" y "*/" se ven como comentarios y son ignoradas por el compilador.

Llamadas al BDOS

La biblioteca estándar de funciones que viene con el compilador C BDS incluye una función para hacer llamadas al BDOS, denominada

"bdos". Toma dos parámetros, y una llamada típica es de la forma siguiente:

```
bdos(c,de);
```

El parámetro "c" representa el valor que será situado en el registro C. Este es el número de código de la función del BDOS. El "de" es el valor que será colocado en el par de registros DE.

La cabecera de biblioteca contiene definiciones (#definición de declaraciones) para las funciones del BDOS 14 a 32, haciendo estas funciones más fáciles de utilizar (figura 11-2, c). La función 32 (obtención/fijación número de usuario en curso) tiene dos definiciones; el parámetro "de" se utiliza para diferenciar si ha de realizarse una función de obtención o de fijación.

Llamadas al BIOS

La biblioteca estándar C BDS contiene también dos funciones que hacen llamadas directas al BIOS. Estas son "bios" y "biosh". Difieren únicamente en que la función bios devuelve el valor al registro A al volver de la rutina del BIOS, mientras que la biosh, como su nombre indica, devuelve el valor al par de registros HL. Ejemplos de su uso son

```
bios(jump_number,bc);
```

y

```
biosh(jump_number,bc,de);
```

Ambas funciones toman como primer parámetro el número de la instrucción de salto del vector de saltos del BIOS al cual hay que transferir el control. Por ejemplo, el punto de entrada de estado de consola es el tercer JMP del vector. Numerando desde 0, éste sería el salto número 2.

El fichero cabecera de biblioteca contiene #definiciones para los saltos del BIOS del 2 al 21 (figura 11-2, d). El último grupo de estas #definiciones (19 a 21) es para las sumas "particulares" a los vectores de salto del BIOS estándar descritos en el capítulo 8.

Recuérdese, sin embargo, que el uso de llamadas directas al BIOS hace más difícil mover los programas de un sistema a otro.

Acceso al bloque de configuración del BIOS

Como se recordará, el bloque de configuración es una colección de estructuras de datos del BIOS. Estas estructuras se utilizan bien para almacenar creaciones reales de ciertas opciones seleccionables por el usuario o para indicar otras estructuras de datos importantes del BIOS.

Uno de los saltos "particulares" añadido al vector de saltos del BIOS transfiere el control a una rutina que devuelve de dirección a la memoria de una estructura de datos específica. Por ejemplo, si un programa de utilidad necesita situar la palabra del BIOS que determina desde qué dispositivo físico hay que leer la entrada de la consola, puede transferir el control al salto 21 del vector de saltos del BIOS (actualmente el salto 22.º) con un valor de código 5 en el registro C. Este salto transfiere el control al código CBGetAddress, que a su regreso fijará el HL en la dirección del vector de redireccionamiento de entrada de consola. El programa de utilidad puede entonces leer de esta variable o escribir en ella. El fichero de cabecera de biblioteca contiene #declaraciones de definición relativas a los valores de código con nombres nemónicos (figura 11-2, e).

Se necesitará recurrir al código fuente de la figura 8-10 para determinar si la dirección devuelta por la función del BIOS es la dirección del elemento de datos o la dirección de una tabla de alto nivel que a su vez señala el elemento de datos.

Para acceder a la fecha actual del sistema, por ejemplo, habrá que incluir el código siguiente:

```
char *ptr_to_date;              /* Declaración de apuntador a fecha */
ptr_to_date = biosh(CB_DATE);   /* Obtención de dirección */
```

El ptr_to_date puede utilizarse entonces para acceder directamente a la fecha.

Durante la depuración inicial de una utilidad es útil poder interceptar todos estos accesos al bloque de configuración, en parte para reasegurarse de que el programa de utilidad está trabajando como debe y en parte para asegurarse de que la rutina del BIOS está devolviendo las direcciones correctas a las estructuras de datos. Sin embargo, la biblioteca de utilidades contiene una función, "get_cba", que obtiene la dirección del bloque de configuración (figura 11-1, a).

Al principio aparece que get_cba se declara como una función que devuelve un apuntador a los caracteres. Esto no es estrictamente cierto. A veces la dirección que devuelve indicará caracteres, otras veces números y a veces estructuras (tales como la de la tabla de teclas de función).

La instrucción "printf" se ha dejado en la función como una anticipación a la utilidad de depuración. Si se necesita ver alguna salida de depuración siempre que se utilice la función get_cba, bórrese el "/*" y el "*/" que rodean a "printf" y recompílese la biblioteca.

Acceso a la tabla de teclas de función del BIOS

El BIOS que se muestra en la figura 8-10 contiene el programa para reconocer cuando una secuencia de escape de entrada indica que se ha

pulsado una de las teclas de función del terminal. En lugar de devolver precisamente la secuencia de escape, el controlador de la consola inyecta una hilera de caracteres previamente programada en la corriente de entrada de la consola. Por ejemplo, en un terminal DEC VT-100, cuando se pulsa la tecla de función PF1, el terminal emite la siguiente secuencia de caracteres: ESCAPE, "O", "P". La tabla de teclas de función contiene el "OP" y una hilera de caracteres terminada en byte 00H para inyectar en la corriente de entrada de la consola. En la figura 8-10, la hilera del ejemplo es "FUNCTION KEY 1", LINE FEED. El fichero de cabeza de biblioteca contiene una declaración para la estructura de la tabla de teclas de función (figura 11-2, h).

Obsérvese el uso de "#define" para declarar la longitud de los caracteres de entrada emitidos por el terminal, así como la longitud de la cadena de salida.

Para acceder a la entrada de una tabla de tecla de función, debe declararse un apuntador a una estructura "_fkt" como ésta:

```
struct _fkt *ptr_to_fkt;       /* Declaración de apuntador */
ptr_to_fkt = get_cba(CB_FKT); /* Fijar apuntador */
printf("Display the first string : %s",
     ptr_to_fkt -> fk_output);
++ptr_to_fkt;                  /* Desplazamiento a entrada siguiente */
```

La función get_cba se utiliza para devolver la dirección de la primera entrada de la tabla de tecla de función y fijar un apuntador en ella. Entonces la función printf (parte de la biblioteca C BDS estándar) se utiliza para imprimir la primera cadena, que viene sustituida por "%s" en la cadena acotada. Obsérvese que la especificación

```
++ptr_to_fkt
```

no añade precisamente uno al apuntador de la tabla de teclas de función —añade todo lo que necesita para desplazar el apuntador a la próxima "entrada" en la tabla.

Acceso a la tabla de dispositivo del BIOS

Las tablas de dispositivo son estructuras importantes para los dispositivos serie servidos por la consola, auxiliares y controladores de dispositivos de listado del BIOS. Se declaran en la línea 1500 de la figura 8-10.

La función get_cba no devuelve un apuntador a una tabla de dispositivo específica, sino un apuntador a una tabla de direcciones de tabla de dispositivo. Cada entrada de la tabla de direcciones corresponde a un número específico de dispositivo. Si no hay tablas de dispositivo para un número específico de éste, entonces la entrada correspondiente de la tabla se

pondrá a cero; el fichero de cabeza de biblioteca contiene definiciones para la tabla de dispositivos (figura 11-2, i).

Las tablas de dispositivo contienen, entre otras cosas, los protocolos de líneas serie actuales utilizadas para sincronizar la transmisión y recepción de datos por medio de los controladores de dispositivos y de los mecanismos físicos. Un ejemplo, PROTOCOL, se muestra más adelante en este mismo capítulo. El ejemplo de definición de declaraciones y definición de estructura que se muestran aquí están modelados para los requerimientos de esta utilidad. Los únicos bytes relevantes son los dos bytes de estado dt_st1 y dt_st2 y la longitud del mensaje utilizado con el protocolo ETX/ACK, dt_etxml. Las definiciones que se muestran son para los bits específicos de los bytes de estado de la tabla de dispositivo. La utilidad PROTOCOL utiliza el bit más significativo para indicar si el establecimiento de un protocolo dado puede coexistir con otros.

Para acceder a estos campos, utilícese el código siguiente:

```
struct _ppdt
     {
     char *pdt[16];    /* Matriz de 16 apuntadores a tablas de dispositivo */
     } *ppdt;          /* Apuntador a la matriz de 16 apuntadores */
struct _dt *dt;        /* Apuntador a la tabla de dispositivo */

ppdt = get_cba(CB_DTA);        /* Fijar apuntador a matriz de apuntadores */
dt = ppdt -> pdt[device_no];   /* Fijar apuntador a tabla de
                                  dispositivo especificado */

if (!dt)
     printf("\nError - no device table for this device.");

dt -> dt_etxml = 0;            /* Puesta a cero de longitud de mensaje ETX */
```

Acceso al bloque de parámetros de disco del BIOS

Algunos de los programas de utilidad mostrados en este capítulo deben acceder al directorio de ficheros en un disco lógico dado. El bloque de parámetros del disco (DPB) indica el tamaño y la situación del directorio de ficheros. La cabeza de biblioteca contiene una definición de la estructura que describe el DPB (figura 11-2, n).

Para situar el DPB, se puede hacer una llamada directa del BIOS a la rutina SELDSK, que devuelve la dirección de la cabecera de parámetros del disco (DPH). Entonces se puede acceder al apuntador DPB del DPH. Alternativamente, usando el BDOS, se puede hacer del disco requerido el disco por defecto y solicitar entonces la dirección de su DPB. El programa para el último método se muestra en la función get_dpb incluida en la biblioteca de utilidades (figura 11-1, u).

La función get_dpb utiliza una función BIOS SELDSK en primer lugar para ver si el disco específico es legítimo. Sólo entonces utilizará el BDOS.

Lectura o escritura de un disco utilizando el BIOS

Cuando se escribe un programa que utiliza llamadas directas al BIOS, se aumenta la posibilidad de problemas al trasladar el programa desde un sistema a otro. Sin embargo, en ciertas circunstancias es necesario utilizar el BIOS. La lectura y la escritura del directorio de ficheros es una de éstas; el BDOS no puede utilizarse para acceder directamente al directorio. La cabecera de la biblioteca contiene una declaración de estructura para un bloque de parámetros que contiene los detalles de una lectura o escritura "absoluta" del disco (figura 11-2, p).

Obsérvese el apuntador a la memoria intermedia de datos de 128 bytes utilizado para mantener uno de los "registros" de CP/M.

Las funciones de lectura y escritura del disco son rd_disk (figura 11-1, k) y wrt_disk (figura 11-1, l). Ambos toman a_drb como parámetros de entrada y ambos llaman a la función set_disk para hacer que el BIOS individual llame a SELDSK, SETTRK y SETSEC.

Hay que observar especialmente el código set_disk (figura 11-1, m) que convierte un sector lógico en un sector físico utilizando la tabla de traducción de sectores y el punto de entrada SECTRAN del BIOS.

Acceso a la entrada del directorio de ficheros

Todas las partes de los programas de utilidad que acceden a un directorio de disco comparten la misma lógica básica sin tener en cuenta su tarea específica. Esta lógica puede describirse de la mejor manera en pseudocódigo:

```
mientras (no al final del directorio)
   {
   acceso a entrada siguiente del directorio
   si (esta entrada coincide con el actual criterio de búsqueda)
      {
      procesamiento de la entrada
      }
   }
```

Existen dos caminos para implementar esta lógica. El primero utiliza el BIOS para leer el directorio. Las entradas se presentan al programa de utilidad exactamente como ocurren en el directorio de ficheros. El segundo utiliza las funciones del BDOS de búsqueda del primero y búsqueda del próximo y accede al directorio fichero a fichero mejor que mediante la entrada. Este último método es más adecuado para utilidades que procesan ficheros más que entradas. la utilidad ERASE, descrita más adelante en este capítulo, ilustra este segundo método.

En la biblioteca están previstos tres grupos de funciones: acceder a la próxima entrada del directorio, buscar coincidencia entre el nombre y la entrada en curso al apretar tecla de búsqueda y asistir con procesamiento al directorio.

Funciones de acceso al directorio

Un cierto número de funciones están relacionadas con el acceso al directorio de ficheros. El primer grupo de tales funciones realiza lo siguiente:

get_nde (obtención de la próxima entrada al directorio; figura 11-1, n)
Esta función devuelve un apuntador a la entrada siguiente del directorio o devuelve cero si se ha alcanzado el final del directorio.

open_dir (apertura de directorio; figura 11-1, o)
Esta función se llama por medio de get_nde para abrir un directorio para procesamiento.

rw_dir (directorio de lectura/escritura, figura 11-1, p)
Esta función lee o escribe el sector del directorio en curso.

err_dir (error del directorio, figura 11-1, q)
Esta rutina de propósito general compone un mensaje de error si el BIOS indica que tiene problemas bien leyendo o escribiendo en el directorio.

Todas estas funciones utilizan un bloque de parámetros de directorio para coordinar su actividad. La cabecera de biblioteca contiene las definiciones para esta estructura (figura 11-2, l), así como definición de declaraciones para códigos de operación utilizados en las funciones de acceso al directorio (figura 11-2, m).

Antes de llamar get_nde, el programa de llamada necesita fijar dp_open a cero (forzando una llamada a open_dir) y el campo de dp_disk al disco lógico correcto. La función open_dir establece todo lo de los campos restantes, utilizando get_dpb para acceder al bloque de parámetros del disco para el disco especificado en dp_disk.

De los restantes indicadores, dp_end se fijará a uno en la realidad, cuando se alcance el final del directorio, y dp_write deberá ser distinto de cero para que rw_dir escriba el sector en curso en el disco.

La función get_nde incluye todo lo concerniente a la lógica necesaria para trasladarse de la entrada de un directorio a la próxima, leyendo en el sector siguiente cuando sea necesario y escribiendo el sector previo si el indicador dp_write ha sido fijado a un valor distinto a cero por el

programa de llamada. Cuenta también el número de entradas del directorio procesadas, detectando e indicando el final del directorio.

El programa al principio de la función llama open_dir si el indicador dp_open es falso. Obsérvese el código al final de open_dir que fija el número de bloques por entrada del directorio (cp_nabpde). Este número está calculado desde el número máximo de bloques del bloque de parámetros del disco. Si es mayor de 255, cada bloque debe ocupar una palabra y habrá ocho bloques por entrada de directorio. Si hay 255 o menos bloques, cada uno tendrá una longitud de un byte y habrá 16 por entrada. El tamaño del bloque en Kbytes se calcula por medio de una sencilla fórmula.

En los primeros pasos de las utilidades de depuración, ha de ponerse un comentario en la línea que realiza la llamada al wrt_disk. Esto prevendrá que se escriba sobre el directorio. Entonces se pueden comprobar también aquellas utilidades que intentan borrar entradas del directorio sin riesgo de dañar cualquier dato del disco.

La última función de este grupo, err_dir, es una función de manejo de errores corriente para cuidar los errores realizados mientras se lee o se escribe el directorio.

Funciones de comparación del directorio

El segundo grupo de funciones que acceden al directorio de ficheros seleccionan por comparación cada entrada del directorio según unos criterios de búsqueda específicos. Estos incluyen las siguientes funciones:

setscb (fijación de bloque de control de búsqueda; figura 11-1, r)
: Un bloque de control de búsqueda (SCB) es una estructura que define las entradas del directorio que hay que seleccionar para procesamiento.

comp_fname (comparación de nombre de fichero; figura 11-1, f)
: Esta función compara el nombre de fichero en la entrada del directorio con una especificada en el bloque de control de búsqueda.

La cabecera de biblioteca contiene la definición de la estructura para el bloque de control de búsqueda (figura 11-2, q). Este SCB es una estructura híbrida. La primera parte de ella es un cruce entre un bloque de control de fichero (FCB) y una entrada de directorio. Los últimos tres campos, scb_length, scb_disk y scb_adisks, son peculiares del bloque de control de búsqueda. Obsérvese que su longitud total es la misma que en el FCB, de forma que puede utilizarse la función estándar C BDS, set_fcb. Esta función fija el nombre y tipo de fichero en un FCB, reemplazando "*" por tantos caracteres "?" como se requiera, y entonces pone a cero todos los bytes no utilizados.

El campo sch_length indica a la función comp_fname (comparación de nombre de fichero) cuántos bytes de la estructura deben ser comparados. Este campo se pondrá a 12 para comparar el número de usuario, el nombre de fichero y el tipo, o a 13 para incluir el número de extensión.

Obsérvese que scb_disk es el disco en curso que ha de buscarse, siempre que scb_adisks sea un mapa de bits con un bit a 1 correspondiendo a cada uno de los 16 posibles discos lógicos que han de ser buscados.

El bloque de control de búsqueda es inicializado por la función setscb.

Obsérvese la forma del nombre de fichero que el setscb espera recibir. Esto se describe en los comentarios al principio de la función.

Algunos programas de utilidad utilizan sus propias versiones especiales de setscb, volviendo a tomar su ssetscb (setscb especial) para evitar que la versión de la biblioteca sea enlazada (linked) con los programas.

La función complementaria comp_fname se utiliza para comparar los primeros bytes de la entrada actual del directorio con los bytes correspondientes del SCB.

La función comp_fname realiza una comparación de cadenas especializada del número de usuario, el nombre y tipo de fichero y, opcionalmente, el número de extensión. Un carácter "?" en el nombre, tipo y extensión de fichero del bloque de control se corresponderá con cualquier carácter de la entrada del directorio. Sin embargo, en el número de usuario del SCB, un "?" corresponderá sólo a un número de valor 0 a 15; no equivaldrá a una entrada del directorio que tenga el byte de número de usuario puesto a E5H (o 0xE5, como notación hexadecimal de C).

Esta función devuelve también uno de varios valores para indicar el resultado de la comparación. Estos valores están definidos en el fichero de cabecera de biblioteca (figura 11-2, j).

Funciones de procesamiento del directorio

El grupo final de funciones que tienen acceso al directorio son aquellas que ayudan al procesamiento de las propias entradas del directorio. Estas funciones utilizan una definición de estructura para acceder a cada entrada del directorio (figura 11-2, o).

Una sentencia de unión se utiliza para los números de bloques. Estos pueden ser entradas de uno o de dos bytes, dependiendo del número máximo de bloques que deban ser representados. La sentencia dice al compilador C del BDS si habrá una matriz de 16 bytes de números enteros cortos (caracteres) o una de ocho enteros de dos bytes sin signo.

Las funciones contenidas en este grupo pueden dividirse en tres subgrupos:

- Aquellos que se enfrentan con la conversión de las entradas del directorio para su composición en la consola.

- Aquellos que se enfrentan con un "mapa de disco" —una matriz conveniente para representar discos lógicos y los números de usuario que contienen.
- Aquellos que se enfrentan con los "vectores de bits" —una representación conveniente de cuáles son los bloques que están en uso o disponibles en un disco lógico.

La biblioteca sólo contiene una función para convertir un nombre de fichero de entrada al directorio en una forma adecuada para componer en la consola. Esta es la función conv_dfname (figura 11-1, h). Toma la información de la entrada del directorio especificada (o, según convenga, un bloque de control de búsqueda) y la formatea en una cadena de la siguiente forma:

```
uu/d:filename.typ
```

El "uu" especifica el número de usuario y el "d" especifica la identificación del disco.

El código repetitivo al final de la función es necesario para asegurar que los caracteres del tipo de fichero no tienen fijados sus bits de mayor orden. Estos bits son los atributos del fichero. Si están fijados, pueden hacer que los caracteres no sean componibles en algunos terminales.

El segundo subgrupo de funciones, aquellas que manipulan un "mapa de disco", producen una matriz que aparece como ésta:

```
Discos
  :
  v  Números de usuario -->                                  -Totales-
  A  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 Utilizados libres
  B
  :
  :
  P
```

Este mapa de disco se utiliza por varios programas de utilidad. Por ejemplo, la utilidad SPACE compone un mapa de disco que muestra, para cada disco lógico del sistema y para cada usuario de cada disco lógico, cuántos Kbytes de espacio del disco están en uso. El total a la derecha muestra el total de espacios usados y libres. En otro ejemplo, la utilidad FIND muestra cuántos ficheros de cada disco y de cada número de usuario coinciden con el nombre de búsqueda.

Cada programa de utilidad que utiliza un mapa de disco se codifica:

```
unsigned disk_map[16][18];
```

Existen dos funciones en la biblioteca para tratar el mapa de disco:

dm_clr (borrado del mapa de disco; figura 11-1, s)
Esta función llena todo el mapa de disco con ceros.

dm_disp (composición del mapa de disco; figura 11-1, t)
Esta función visualiza las líneas de título horizontal y vertical para el mapa de disco y convierte entonces cada elemento del mapa de disco en un número decimal.

La primera función, dm_clr, utiliza una de las funciones C estándar del BDS para fijar un bloque de memoria a un valor específico. Considera que el mapa de disco es de 16×18 elementos, cada uno de dos bytes de longitud.

La segunda función, dm_disp, imprime líneas horizontales sólo para aquellos discos especificados en el parámetro del mapa de bits. Aquí se muestra un ejemplo de su salida:

```
      0   1   2   3   4   ...   10  11  12  13  14  15 Utilizados libres
A:    1   1                                                  15       241
B:   66  20  74  50   3                                      245       779
C:   -- Ninguno --                                            0      1024
(NOTA: Todos los grupos de usuarios se mostrarán en el terminal.)
```

El último subgrupo se enfrenta con el procesamiento de "vectores de bits". Un vector de bits es una hilera de bits empaquetada en bloques de ocho bits por byte. Cada bit está direccionado por su número correspondiente en el vector; el primer bit es el número 0.

Un ejemplo de por qué se utilizan los vectores de bits es un programa de utilidad que necesite examinar el directorio de un disco y construir una estructura que muestre cuáles son los bloques que están en uso. Puede hacerlo accediendo a cada elemento activo del directorio y, para cada número de bloque distinto a cero, situando el número de bit correspondiente en un vector de bits.

La cabecera de biblioteca tiene una definición de estructura para un vector de bits (figura 11-2, s). Este vector contiene la longitud total del vector de bits en bytes y dos apuntadores. El primero señala el comienzo del vector, el segundo el final. Los bytes que contienen los propios bits del vector están asignados por la función alloc —una de las funciones C estándar del BDS.

Las siguientes funciones de vector de bits están previstas en la biblioteca:

bv_make (realizar vector de bits; figura 11-1, cc)
Esta función asigna memoria para el vector de bits (utilizando el mecanismo estándar del BDS C) y fija todos los bits a cero.

bv_fill (llenado del vector de bits; figura 11-1, dd)
Esta llena un vector específico, fijando cada byte en un valor especificado.

bv_set (fijación del vector de bits; figura 11-1, ee)
Esta fija a uno el bit especificado de un vector.

bv_test (comprobación del vector de bits; figura 11-1, ff)
Esta función devuelve un valor cero o uno, reflejando la fijación del bit especificado de un vector de bits.

bv_nz (vector de bits distinto de cero; figura 11-1, gg)
Esta devuelve cero o un valor distinto de cero para reflejar si hay *algún* bit a uno en el vector de bits especificado.

bv_and (vector de bits AND; figura 11-1, hh)
Esta función realiza un AND booleano entre dos vectores de bits y coloca el resultado en un tercer vector.

bv_or (vector de bits OR; figura 11-1, ii)
Esta es similar a la bv_and, excepto que realiza un OR inclusivo de los vectores de entrada.

bv_disp (visualización de vector de bits; figura 11-1, jj)
Esta función compone una línea de título e imprime entonces los contenidos del vector de bits especificado como una serie de ceros y unos. Cada bit está formateado para hacer más fácil de leer la salida.

La función bv_make utiliza la función alloc para asignar un bloque de la parte no utilizada de la memoria entre el final de un programa y la base del BDOS. Requiere que dos estructuras de datos sean declaradas al principio del programa. Estas estructuras se declaran en el fichero de la cabecera de biblioteca (figura 11-2, b).

La función bv_fill utiliza la función setmen del BDS C estándar.

La función bv_set convierte el número de bit en un desplazamiento sobre el byte desplazando el número de bit tres lugares a la derecha. Los tres bits menos significativos del número de bit original especifican qué bit del byte adecuado necesita ser operado (OR).

La función bv_test es efectivamente el reverso de bv_set. Accede al bit específico y devuelve su valor al programa que llama.

La función bv_nz registra todo el vector de bits buscando el primer byte distinto de cero. Si todo el vector es cero, devuelve un valor cero. En otro caso, devuelve un apuntador al primer byte distinto de cero.

Las dos funciones bv_and y bv_or toman tres vectores de bits como parámetros. El primer vector se utiliza para contener el resultado tanto de operar con AND o con OR el segundo y el tercer vector. Ambas funciones suponen que el vector de salida ya ha sido creado utilizando bv_make. El

más corto de los tres vectores terminará la función bv_and o bv_or; es decir, estas funciones terminarán cuando se alcance el final del primer vector (el más corto).

La última función, bv_disp, compone la línea de título especificada por el programa de llamada, y entonces compone todos los bits del vector, con el número de bit del primer bit de cada línea mostrado a la izquierda.

Ninguno de los programas de utilidad usa bv_disp —se ha dejado en la librería simplemente como ayuda para depuración.

Aquí hay un ejemplo de la salida del bv_disp:

```
Vector de bits : Bloques en uso
   0 : 0000 0000  0001 1000  1000 0001  1111 1111  1111 1111
  40 : 1111 1111  1111 1111  1111 1111  1110 1011  0000 0000
  80 : 1100 0000  1111 1100  1111 1001  1100 0000  1001 1111
 120 : 1110 1100  0001 1111  0000 0000  1101 1000  0001 1110
 160 : 1111 1111  1110 1111  1110 1111  0000 0111  0000 0111
 200 : 1111 0010
```

Comprobación de parámetros especificados por el usuario

El lenguaje C proporciona un mecanismo para acceder a los parámetros especificados en la zona de parámetros de orden. Proporciona una cuenta del número de parámetros introducidos, "argc" (cuenta de argumentos), y una matriz de apuntadores para cada una de las cadenas de caracteres, "argv" (vector de argumentos). Al principio de la función principal de cada programa se deben definir estas dos variables como sigue:

```
main(argc,argv)
{
int argc;       /* Contador de argumentos */
char *argv[];  /* Matriz de apuntadores a cadenas de caracteres */
:
: /* Resto de la función principal */
:
}
```

Consideremos el caso mínimo —una línea de órdenes únicamente con el nombre del programa:

```
A>command
```

El convenio es que el primer argumento de la línea es el nombre del propio programa. Por tanto, argc se pondrá a uno y argv[0] será un apuntador al nombre del programa, "orden".

Ahora consideraremos un caso más complejo —una línea de órdenes con parámetros como el siguiente:

```
A>command param1 123
```

En este caso, argc será tres; argv[1] será un apuntador a paraml; y argv[1][0] accederá al carácter 0 (el primero) de argv[1] —en este caso el carácter "p".

Para detectar si el segundo parámetro está presente y es numérico, el programa será:

```
if (isdigit(argv[1][0]))
     {
          /* Procesamiento de dígito */
     }
else
     {
          /* Parámetro no presente o con carácter
            alfabético en primer lugar */
     }
```

En la mayoría de las utilidades, se obtendrá un programa mucho más "amistoso" si el usuario necesita sólo caracteres específicos suficientes de un parámetro para distinguir el valor introducido de otros valores posibles. Por ejemplo, consideremos un programa que pueda tener como parámetro uno de los siguientes valores: 300, 600, 1200, 2400, 4800, 9600 ó 19200. Sería conveniente que el usuario necesitara mecanografiar únicamente el primer número, mejor que tener que introducir rayas enteras de claves redundantes. Sin embargo, los valores 1200 y 19200 serían entonces ambiguos. El usuario tendría que introducir 12 ó 19. Los usuarios novatos prefieren a menudo especificar los parámetros completos para claridad y seguridad.

La biblioteca C estándar proporciona una función de comparación de cadenas de caracteres, strcmp. Desgraciadamente, esta función no prevé la comparación descrita anteriormente. Sin embargo, la biblioteca posee dos funciones especiales que lo hacen posible: sstrcmp (comparación de subcadenas, figura 11-1, d) y usstrcmp (comparación de subcadenas en mayúsculas, figura 11-1, e). La última función es necesaria cuando se necesita comparar una subcadena que pueda contener caracteres en minúscula, y convierte los caracteres en mayúsculas antes de la comparación.

Para asistir a la manipulación de cadenas de caracteres, se han incluido dos funciones adicionales en la biblioteca. Estas son strscn (examen de cadenas, figura 11-1, b) y ustrcmp (comparación de cadenas en mayúsculas, figura 11-1, c).

Utilización de tablas de código

Una tabla de código es una simple estructura usada por todos los programas de utilidad que aceptan parámetros que pueden tener cualquiera de varios valores. La cabecera de biblioteca contiene una definición de estructura para tablas de código (figura 11-2, r).

Una entrada de tabla de código contiene un valor de código sin signo y un apuntador a una cadena de caracteres. Se utiliza en los programas de utilidad siempre que hay necesidad de relacionar algún número de código arbitrario o una configuración de bits con una hilera de caracteres ASCII. Por ejemplo, para programar un chip generador de relación de velocidad de transmisión (baud-rate) a varios valores requiere diferentes constantes de tiempo para cada valor. Los usuarios no necesitan saber cuáles son estos números; sólo necesitan estar preparados para especificar el valor de la velocidad de transmisión como una cadena ASCII.

Por tanto, una tabla de códigos se fija como sigue:

Constante de velocidad de transmisión	Nombre de usuario
0x35	"300"
0x36	"600"
0x37	"1200"
0x3A	"2400"
0x3C	"4800"
0x3E	"9600"
0x3F	"19200"

Un programa de utilidad necesita ahora estar preparado para realizar varias operaciones utilizando la tabla de código:

- Dado el parámetro de entrada en la zona de parámetros de la orden, la utilidad debe comprobar si la cadena ASCII está en la tabla de código, componiendo todas las opciones legales en la consola si no está, y si está, devolviendo el valor del código para procesos subsiguientes.
- Dada la constante actual (contenida en el BIOS), la utilidad deberá registrar la tabla de códigos y componer la cadena ASCII correspondiente para decir al usuario su valor baud actual.

La biblioteca incluye funciones especializadas para hacer esto, más algunas funciones adicionales para hacer más generalmente utilizables las tablas de códigos. Estas funciones son:

ct_init (inicialización de tabla de códigos; figura 11-1, v)
Esta función inicializa una entrada específica de una tabla de códigos, fijando el valor del código y el apuntador a la cadena de caracteres.

ct_parc (código de retorno de parámetro de tabla de códigos; figura 11-1, w)
Esta realiza la comparación entre una subcadena de mayúsculas con una cadena de clave especificada, devolviendo bien un error (el valor CT_SNF —hilera no encontrada—) o bien un valor de código.

ct_code (código de retorno de tabla de códigos; figura 11-1, x)
Esta función es similar a ct_parc en que examina la tabla de códigos y devuelve el código correspondiente. Difiere en la forma de realizar la comparación. Se compara la cadena completa de búsqueda con la cadena de la entrada de la tabla de códigos. Existe una concordancia sólo cuando todos los caracteres son iguales.

ct_disps (visualización de cadenas de tabla de códigos; figura 11-1, y)
Esta función visualiza todas las cadenas en una tabla de códigos dada. Se utiliza bien cuando el usuario ha introducido una cadena no válida o cuando el programa de utilidad requiere que se muestren las opciones que están disponibles para un parámetro.

ct_index (retorno de índice de tabla de códigos; figura 11-1, z)
Dada una cadena, esta función busca en la tabla de códigos y devuelve el *índice* de la entrada que tiene una cadena que coincide con la de búsqueda. El índice no es el valor del código; es el número de la entrada de la tabla.

ct_stri (índice de cadena de tabla de códigos; figura 11-1, aa)
Dado un número de índice de entrada, esta función devuelve un apuntador a la cadena en aquella entrada.

ct_strc (código de cadena de tabla de códigos; figura 11-1, bb)
Dado un número de código, esta función devuelve un apuntador a la cadena en la entrada que tiene un número de código que coincide.

Acceso a un directorio por medio del BDOS

Un problema asociado al acceso directo al directorio de ficheros, como se ha ilustrado en funciones anteriores, es que el programa se presente con entradas de directorio en el orden exacto en que ocurren en él. Para algunos programas, tales como aquellos que procesan grupos de ficheros, es mejor utilizar las funciones de búsqueda primera y de búsqueda del siguiente del BDOS para acceder al directorio.

Utilizando el BDOS, el programa puede procesar el nombre del primer fichero para confrontarlo con la clave de búsqueda ambigua, volver después al BDOS para obtener el nombre del fichero siguiente, etc. La cabecera de biblioteca contiene una definición de estructura para un bloque de control de ficheros CP/M (figura 11-2, k).

Obsérvese que el primer byte del FCB es un número de disco más que el número de usuario de la entrada del directorio. Obsérvese también el uso de una sentencia para describir los números de bloque.

La biblioteca estándar del C BDS contiene una función, setfcb, a la que se ha dado la dirección de un FCB y un apuntador a una cadena que

contiene un nombre de fichero. Convierte cualquier "*" en el nombre en el número adecuado de "?" y llena el resto del FCB con ceros.

La biblioteca ejemplificada contiene las funciones siguientes, diseñadas para el acceso al directorio de ficheros del BDOS:

get_nfn (obtención del nombre de fichero siguiente; figura 11-1, i)
Esta función posee un apuntador a un nombre de fichero ambiguo y un apuntador a un FCB. Vuelve con el FCB fijado para acceder al próximo fichero que coincida con el nombre ambiguo.

srch_file (búsqueda de fichero; figura 11-1, j)
Esta función, utilizada por get_nfn, lanza tanto una llamada de búsqueda primera como una llamada de búsqueda del siguiente del BDOS.

conv_fname (conversión del nombre de fichero; figura 11-1, g)
Esta función convierte un nombre de fichero de un FCB en una forma adecuada para visualización en la consola. Es similar a la función conv_dfname, descrita anteriormente, excepto que envía sólo el disco, nombre de fichero y tipo (no el número de usuario) en la forma siguiente:

```
d:filename.typ
```

Para señalar a la función get_nfn que se desea el primer nombre de fichero, hay que fijar el bit más significativo del primer byte, el número de disco.

Aquí se muestra un ejemplo de cómo usar la función get_nfn:

```
struct _fcb fcb;              /* Declaración de bloque de control de fichero */

setmem(fcb,FCB_SIZE,0);       /* Puesta a cero del FCB */
fcb.fcb_disk = 0x80;          /* Marcar el FCB para "primera vez" */

while (get_nfn(fcb,"B:XYZ*.*"))
                              /* Hasta que get_nfn devuelve cero */
     {
                              /* Apertura de fichero utilizando el FCB */
     while                   (/* No al final del fichero */)
          {
                              /* Procesamiento del registro siguiente o
                                   carácter en fichero */
          }
                              /* Cerrar fichero */
     }
```

La cadena entre comillas "B:XYZ*.*" podría ser también sólo un apuntador a cadena o un parámetro de la línea de orden, argv[n].

La última función para el proceso del directorio de ficheros del BDOS, conv_fname, se utiliza para convertir un nombre de fichero para salida por un terminal. De nuevo, el código repetitivo al final borra los bits de

atributos de fichero para evitar cualquier efecto secundario desde el terminal.

Programas de utilidad que mejoran el CP/M estándar

Este grupo de utilidades se ha diseñado para mejorar aquellos suministrados por Digital Research. No gozan de las ventajas del BIOS mejorado de la figura 8-10 y pueden utilizarse en *cualquier* instalación con la versión 2.2 del CP/M.

Con la excepción de la utilidad ERASE, todas las utilidades registran el directorio de ficheros utilizando llamadas al BIOS, como se describe anteriormente en este mismo capítulo.

ERASE—Una forma segura de borrar ficheros

Existen dos inconvenientes en el procesador de órdenes de consola construido con la orden ERA. En primer lugar, borrará indiscriminadamente grupos de ficheros. En segundo lugar, si se tiene un nombre de fichero con caracteres no gráficos o en minúsculas, no se puede usar la orden ERA, porque el CCP convierte los caracteres de los parámetros de la orden a mayúsculas y da por finalizado un nombre de fichero al encontrar cualquier carácter extraño en la cadena.

La utilidad ERASE que se muestra en la figura 11-3 borra grupos de ficheros, pero pide al usuario la confirmación antes de borrar cada uno de ellos.

Mejor que utilizar el BIOS para acceder a la entrada de cada directorio, utiliza la función get_nfn, que llama entonces al BDOS. Por tanto, ERASE funciona igualmente bien para ficheros que tienen múltiples entradas en el directorio. Puede utilizar la función de borrado de ficheros del BDOS (Delete) para borrar todas las extensiones de un fichero dado.

A continuación se da un ejemplo de diálogo de consola que muestra ERASE en operación:

```
P3A>erase<CR>
ERASE Version 1.0 02/23/83 (Library 1.0)
Usage :
        ERASE {d:}file_name.typ


P3A>erase *.com<CR>
ERASE Version 1.0 02/23/83 (Library 1.0)

Búsqueda de fichero que coincide con A:????????.COM
        Erase A:UNERASE .COM y/n? n
```

```
Erase A:TEMP1    .COM y/n? y <== ¡Será borrado!
Erase A:TEMP2    .COM y/n? n
Erase A:TEMP3    .COM y/n? n
Erase A:TEMP4    .COM y/n? y <== ¡Será borrado!
Erase A:ERASE    .COM y/n? n

Borrado de ficheros...
File A:TEMP1    .COM erased.
File A:TEMP4    .COM erased.
```

```
#define VN "1.0 02/24/83"

/* ERASE
/* Esta función borra lógicamente el fichero(s) especificado utilizando la
   función delete de BDOS. */

#include <LIBRARY.H>

struct _fcb amb_fcb;              /* Bloque de control de fichero con nombre ambiguo */
struct _fcb fcb;                  /* Usado por las funciones de búsqueda del BDOS */

char file_name[20];               /* Formato para visualizar d:Nombre-de-fichero.tipo */
short cur_disk;                   /* Disco lógico en curso al principio del programa */
                                  /* ERASE salva el FCB de todos los ficheros
                                     que serán borrados en la matriz
                                     siguiente  */
#define MAXERA 1024
struct _fcb era_fcb[MAXERA];
int ecount;                       /* Cuenta del número de ficheros a borrar */
int count;                        /* Utilizado para acceso a era_fcb durante el borrado */


main(argc,argv)
short argc;               /* Cuenta de argumentos */
char *argv[];             /* Vector de argumentos (apuntador a una matriz de caracteres) */
{

printf("\nERASE Version %s (Library %s)",VN,LIBVN);
chk_use(argc);                    /* Comprobación de uso */
cur_disk = bdos(GETDISK);         /* Obtención de disco por defecto en curso */

ecount = 0;                       /* Inicialización de cuenta de ficheros a borrar */

setfcb(amb_fcb,argv[1]);          /* Fijación de nombre de fichero ambiguo */
if (amb_fcb.fcb_disk)             /* Comprobación de si se utiliza el disco por defecto */
        {
        bdos(SETDISK,amb_fcb.fcb_disk + 1);      /* Fijación del disco especificado */
        }
        /* Conversión del nombre de fichero ambiguo para salida */
conv_fname(amb_fcb,file_name);
printf("\n\nSearching for file(s) matching %s.",file_name);

        /* Fijación del bloque de control de fichero para indicar primera búsqueda */
fcb.fcb_disk |= 0x80;    /* OR en el bit más significativo */

        /* Mientras no se llegue al final del fichero, fijar el FCB al nombre
           siguiente que coincida */
while(get_nfn(amb_fcb,fcb))
        {
        conv_fname(fcb,file_name);
                /* Pregunta de borrado o no de fichero */
        printf("\n\tErase %s y/n? ",file_name);
        if (toupper(getchar()) == 'Y')
                {
                printf(" <== Will be erased!");
                        /* Añadir FCB en curso a matriz de FCB's */
                movmem(fcb,&era_fcb[ecount++],FCBSIZE);
                        /* Comprobación de tabla no completa */
```

Figura 11-3. ERASE.C, una utilidad que solicita confirmación antes de borrar.

```
                if (ecount == MAXERA)
                        {
                        printf("\nWarning : Internal table now full. No more files can be erased")
                        printf("\n   until those already specified have been erased.");
                        break;  /* Salto fuera del bucle mientras */
                        }
                }
        }       /* Procesadas todas las entradas al directorio */

if (ecount)
        printf("\n\nErasing files now...");

        /* Procesamiento de cada uno de los FCB's en la matriz, borrando los ficheros */
for (count = 0;          /* Comienzo con el primer fichero en la matriz */
     count < ecount;     /* Hasta procesamiento de todas las entradas activas */
     count++)            /* Desplazamiento hasta el FCB siguiente */
        {
        conv_fname(&era_fcb[count],file_name);
        if (bdos(DELETEF,&era_fcb[count]) == -1)        /* ¿Error? */
                printf("\n\007Error trying to erase %s",file_name);
        else            /* Fichero borrado */
                printf("\n\tFile %s erased.",file_name);
        }
bdos(SETDISK,cur_disk);  /* Puesta a disco en curso */
}


chk_use(argc)           /* Comprobación de uso */
/* Esta función comprueba que se haya especificado el número correcto
   de parámetros, avisando instrucciones en caso contrario. */


/* Parámetros de entrada */
int argc;       /* Cuenta del número de argumentos en línea de orden */

{
        /* El valor mínimo de argc es 1 (por el propio nombre del programa);
           por tanto, argc es siempre mayor en una unidad al número
           de parámetros de la línea de orden */

if (argc != 2)
        {
        printf("\nUsage :");
        printf("\n\tERASE [d:]file_name.typ");
        exit();
        }

}
```

Figura 11-3. (Continuación.)

UNERASE—Recuperación de ficheros borrados

UNERASE, como su nombre indica, puede utilizarse para "resucitar" un fichero borrado accidentalmente. Sólo los ficheros cuyos bloques no han sido asignados nuevamente a otros ficheros pueden resucitarse. La utilidad UNERASE mostrada en la figura 11-4 construye un vector de bits de todos los bloques utilizados por las entradas activas del directorio. Entonces construye un vector de bits de todos los bloques requeridos por el fichero a recuperar (UNERASE'd). Si un AND booleano entre los dos vectores produce un vector distinto de cero, entonces uno o más de los bloques que originalmente pertenecían al fichero borrado se encuentran ahora asignados a otros ficheros del disco.

```
#define VN "1.0 02/12/83"

/* UNERASE --
   Esta utilidad es la inversa de ERASE. Restaura
   al directorio los ficheros especificados cambiando el primer
   byte de su entrada al directorio (0xE5) al número de usuario
   especificado. */

#include <LIBRARY.H>

struct _dirpb dir_pb;              /* Bloque de parámetros de manejo del directorio */
struct _dir *dir_entry;            /* Apuntador a la entrada del directorio */
struct _scb scb;                   /* Bloque de control de búsqueda */
struct _scb scba;                  /* SCB activado para coincidencia con todos los ficheros */
struct _dpb dpb;                   /* Bloque de parámetros de disco de CP/M */
struct _bv inuse_bv;               /* Vector de bits de bloques utilizados */
struct _bv file_bv;                /* Vector de bits para fichero a recuperar */
struct _bv extents;                /* Vector de bits para extensiones a recuperar */

char file_name[20];                /* Formateado para visualizar un/de:Nombre-de-fichero.tipo */

short cur_disk;                    /* Disco lógico en curso al principio del programa
                                      NZ = mapa mostrado del número de ficheros */
int count;                         /* Utilizando para acceso al número de bloques en cada
                                      entrada del directorio */
int user;                          /* Usuario sobre el que se recupera el fichero */


main(argc,argv)
short argc;              /* Cuenta de argumentos */
char *argv[];            /* Vector de argumentos (apuntador a matriz de caracteres) */

{
printf("\nUNERASE Version %s (Library %s)",VN,LIBVN);
chk_use(argc);                     /* Comprobación de uso */
cur_disk = bdos(GETDISK);          /* Comprobación de disco por defecto en curso */

        /* Utilizando una versión especial de la utilidad bloque de control
           de búsqueda fijar, disco, nombre, tipo (no ambiguos) y número de usuario
           para comparar únicamente entradas suprimidas y longitud para comparar
           usuario, nombre y tipo.
           Esta versión especial devuelve también el disk_id tomado
           del nombre de fichero en la linea de orden.  */
if ((dir_pb.dp_disk = ssetscb(scb,argv[1],0xE5,12)) == 0)
        {       /* Utilización del disco por defecto */
        dir_pb.dp_disk = cur_disk;
        }
else
        {       /* Disco A = 0, B = 1 (para SELDSK) */
        dir_pb.dp_disk--;
        }
printf("\nSearching disk %d.",dir_pb.dp_disk);

if(strscn(scb,"?"))        /* Comprobación de nombre ambiguo */
        {
        printf("\nError -- UNERASE can only revive a single file at a time.");
        exit();
        }

        /* Activación de un bloque de control de búsqueda especial que contrasta
           con todos los ficheros existentes. */

ssetscb(scba,"*.*",'?',12);        /* Fijar nombre de fichero e inicializar SCB */

if (argc == 2)                     /* Número de usuario no especificado */
        user = bdos(GETUSER,0xFF);       /* Obtención de número de usuario en curso */
else
{
        user = atoi(argv[2]);            /* Obtención de número especificado */
        if (user > 15)
                {
                printf("\nUser number can only be 0 - 15.");
                exit();
                }
}
```

Figura 11-4. UNERASE.C, un programa de utilidad que "resucita" ficheros borrados.

```
/* Construcción de un vector de bits que muestra los bloques en uso
   actualmente. SCB se ha activado para contrastar con todas las entradas
   activas del directorio en el disco. */
build_bv(inuse_bv,scba);

/* Construcción de un vector de bits para el fichero a recuperar
   mostrando qué bloques se precisan para él. */
if (!build_bv(file_bv,scb))
        {
        printf("\nNo directory entries found for file %s.",
                argv[1]);
        exit();
        }
/* Realización de la op. (AND) entre los dos vectores de bits. */
bv_and(file_bv,inuse_bv,file_bv);

/* Comprobación de resultado distinto de cero -- si es asi, uno o
   más bloques requeridos por el fichero borrado están
   actualmente usados para un fichero existente y el fichero no
   puede ser recuperado. */
if (bv_nz(file_bv))
{
        printf("\n--- This file cannot be restored as some parts of it");
        printf("\n    have been re-used for other files! ---");
        exit();
}


/* Continuación de la recuperación del fichero cambiando todas las entradas
   al directorio para contener el número de usuario especificado.
   Nota: Para el mismo nombre y tipo de fichero pueden existir
   varias entradas al directorio y hasta con el mismo número de
   extensión. Por esta razón se mantiene un mapa de bits
   de los números de extensión recuperados -- No serán recuperados números
   de extensión duplicados. */

/* Fijar el vector de bits para más de 127 extensiones recuperadas */
bv_make(extents,16);            /* 16 * 8 bits */

/* Fijación del directorio a cerrado y forzar la función get_nde
   para abrirlo. */
dir_pb.dp_open = 0;

/* Mientras no se llegue al final del directorio devolución de apuntador
   a entrada siguiente. */
while(dir_entry = get_nde(dir_pb))
{

        /* Comprobación de coincidencia de nombre, tipo y usuario = 0xE5 */
if (comp_fname(scb,dir_entry) == NAME_EQ)
        {
                /* Comprobación de si esta extensión ha sido
                   ya recuperada */
        if (bv_test(extents,dir_entry -> de_extent))
                {                       /* Si, lo ha sido */
                printf("\n\t\tExtent #%d of %s ignored.",
                        dir_entry -> de_extent,argv[1]);
                continue;               /* No recuperarla */
                }
        else                            /* Indicación de extensión recuperada */
                {
                bv_set(extents,dir_entry -> de_extent);
                dir_entry -> de_userno = user;  /* Entrada recuperada */
                dir_pb.dp_write = 1;    /* Necesidad de escritura en sector anterior */
                printf("\n\tExtent #%d of %s unerased.",
                        dir_entry -> de_extent,argv[1]);
                }
        }
}

printf("\n\nFile %s unerased in User Number %d.",
argv[1],user);

bdos(SETDISK,cur_disk);  /* Puesta a disco en curso */
}
```

Figura 11-4. (Continuación.)

```
build_bv(bv,scb)            /* Construcción del vector de bits (del directorio) */
/* Esta función examina el directorio del disco especificado en el bloque
   de parámetros del directorio (declarado como variable global)
   y construye el vector de bits especificado mostrando todos los bloques
   utilizados por ficheros que coinciden con el nombre en el bloque
   de control de búsqueda. */

/* Parámetros de entrada */
struct _bv *bv;              /* Apuntador al vector de bits */
struct  scb *scb;            /* Apuntador al bloque de control de búsqueda */
/* Utiliza también el bloque de parámetros del directorio */

/* Parámetros de salida
   Se creará el vector de bits especificado que tendrá a 1 los bits
   correspondientes a bloques encontrados en una entrada del directorio
   que coincide con el bloque de control de búsqueda.
   Devuelve también el número de entradas coincidentes. */
{
unsigned abno;               /* Número de bloque */
struct _dpb *dpb;            /* Apuntador al bloque de parámetros del disco en el BIOS */
int mcount;                  /* Cuenta de entradas al directorio coincidentes */

mcount = 0;                  /* Inicialización de cuenta */
dpb = get_dpb(dir_pb.dp_disk); /* Obtención de dirección del bloque de parámetros del disco */

/* Se hace el vector de bits con un byte por cada ocho
   bloques + 1 */
if (!(bv_make(bv,(dpb -> dpb_maxabn >>3)+1)))
        {
        printf("\nError -- Insufficient memory to make a bit vector.");
        exit();
        }


/* Fijación del directorio a "cerrado" para forzar la función
   get_nde para abrirlo. */
dir_pb.dp_open = 0;

/* Examen del directorio construyendo el vector de bits */
while(dir_entry = get_nde(dir_pb))
        {
                /* Comparación de número de usuario (que puede ser 0xE5),
                   nombre y tipo de fichero. */
        if (comp_fname(scb,dir_entry) == NAME_EQ)
                {
                ++mcount;                   /* Actualización cuenta de coincidencias */
                for (count = 0;             /* Comienzo con el primer bloque */
                    count < dir_pb.dp_nabpde;   /* Para número de bloques por entrada al directorio */
                    count++)
                        {
                                /* Puesta a uno del bit apropiado para cada número
                                   de bloque distinto de cero */
                        if (dir_pb.dp_nabpde == 8)       /* Suposición de 8 números de 2 bytes */
                                {
                                abno = dir_entry -> _dirab.de_long[count];
                                }
                        else    /* Suposición de 1 número de 16 bytes */
                                {
                                abno = dir_entry -> _dirab.de_short[count];
                                }
                        if (abno) bv_set(bv,abno); /* Puesta a uno del bit */
                        }
                }
        }
return mcount;               /* Devolución del número de entradas coincidentes */
}


chk_use(argc)                /* Comprobación de uso */
/* Esta función comprueba que se haya especificado el número
   correcto de parámetros, avisando instrucciones en caso
   contrario. */

/* Parámetros de entrada */
int argc;           /* Cuenta del número de argumentos en línea de orden */
{
```

Figura 11-4. (Continuación.)

```
/* El valor mínimo de argc es 1 (por el propio nombre del programa);
   por tanto, argc es siempre mayor en una unidad al número
   de parámetros de la línea de orden */

if (argc == 1 || argc > 3)
        {
        printf("\nUsage :");
        printf("\n\tUNERASE [d:]filename.typ [user]");
                printf("\n\tOnly a single unambiguous file name can be used.)");
        exit();
        }
} /* Final de chk_use */


ssetscb(scb,fname,user,length)   /* Versión especial del bloque de control de búsqueda */
/* Esta función fija el bloque de control de búsqueda
   de acuerdo con el nombre de fichero
   especificado.
   El nombre de fichero puede presentarse en las formas siguientes:
        Nombre-de-fichero
        Nombre-de-fichero.tipo
        d:nombre-de-fichero.tipo

   La función liga el mapa de bits de acuerdo con los discos en los que
   se busca. Para cada uno de los discos relacionados, comprueba si se
   ha generado algún error en la selección del disco (es decir, si existen
   las tablas de disco en el BIOS para él). */

/* Parámetros de entrada */
struct _scb *scb;          /* Apuntador al bloque de control de búsqueda */
char *fname;               /* Apuntador al nombre de fichero */
short user;                /* Número de usuario a buscar */
int length;                /* Número de bytes a comparar */

/* Parámetros de salida
   Número de disco buscado (A = 1, B = 2...)
*/
{
short disk_id;             /* Número de disco a buscar */

setfcb(scb,fname);         /* Fijación del bloque de control de búsqueda como si
                              se tratara de un bloque de control de fichero. */
disk_id = scb -> scb_userno;     /* Fijación de disk_id antes de que sea escrito
                                    por el número de usuario */
scb -> scb_userno = user;        /* Fijación del número de usuario */
scb -> scb_length = length;      /* Fijación del número de bytes a comparar */
return disk_id;
} /* Final de setscb */
```

Figura 11-4. (Continuación.)

Una complicación más ocurre si dos o más entradas del directorio del fichero borrado tienen el mismo número de extensión. Esto puede ocurrir si el fichero se ha creado y borrado varias veces. Bajo estas circunstancias, UNERASE recupera la primera entrada con un número de extensión dado que encuentra, y compone un mensaje en la consola tanto cuando se recupera una extensión como cuando se ignora.

Debido a la naturaleza complicada del proceso de UNERASE, esta utilidad puede procesar sólo un nombre único y no ambiguo de fichero.

El siguiente diálogo de consola muestra a UNERASE en operación:

```
P3A>dir *.com<CR>
A: UNERASE  COM : TEMP2    COM : TEMP3    COM : ERASE    COM
```

```
P3A>unerase<CR>
UNERASE Version 1.0 02/12/83 (Library 1.0)
Usage :
        UNERASE {d:}filename.typ {user}
        Only a single unambiguous file name can be used.

P3A>unerase temp1.com<CR>
UNERASE Version 1.0 02/12/83 (Library 1.0)
Searching disk A.
        Extent #0 of TEMP1.COM unerased.
                Extent #0 of TEMP1.COM ignored.

File TEMP1.COM unerased in User Number 3.

P3A>dir *.com<CR>
A: UNERASE  COM : TEMP1    COM : TEMP2    COM : TEMP3    COM
A: ERASE    COM

P3A>unerase temp5.com<CR>
UNERASE Version 1.0 02/12/83 (Library 1.0)
Searching disk A.
No directory entries found for file TEMP5.COM.
```

FIND—Encuentro de ficheros "perdidos"

La utilidad FIND mostrada en la figura 11-5 busca todos los números de usuario en los discos lógicos especificados, comparando cada entrada con un nombre de fichero ambiguo. Entonces puede visualizar bien un mapa de disco que muestre cuántos ficheros concordantes se han encontrado en cada número de usuario para cada disco o el número de usuario, el nombre del fichero y el tipo para cada entrada de directorio comparada.

Se puede utilizar FIND para localizar un fichero específico o un grupo de ficheros, como se muestra en el siguiente diálogo de consola:

```
P3B>find<CR>
FIND Version 1.0 02/11/83 (Library 1.0)
Usage :
        FIND d:filename.typ {NAMES}
             *:filename.typ (All disks)
             ABCD..OP:filename.typ (Selected Disks)
        NAMES option shows actual names rather than map.

P3B>find ab:*.*<CR>
FIND Version 1.0 02/11/83 (Library 1.0)
Searching disk : A
Searching disk : B
                Numbers show files in each User Number.
                          --- User Numbers ---          Dir. Entries
      0   1   2   3   4   5   ...   11  12  13  14  15  Used Free
A:    1   1       8                                      23  233
B:   66  20  74  55   3                                  252  772

P3B>find *:*.com<CR>
FIND Version 1.0 02/11/83 (Library 1.0)
Searching disk : A
Searching disk : B
Searching disk : C
```

```
                                  --- User Numbers ---          Dir. Entries
      0    1    2    3    4    5    ...    11   12   13   14   15  Used Free
A:                  5                                                23  233
B:   61    5    4   13                                               252  772
C:   -- None --                                                       16  112


P3B>find *.com names<CR>
FIND Version 1.0 02/11/83 (Library 1.0)
Searching disk : B
 0/B:CC       .COM  0/B:CC2      .COM  0/B:CLINK    .COM  2/B:CLIB    .COM
 1/B:CPM61    .COM  1/B:MOVCPM   .COM  1/B:PSWX     .COM  0/B:SUBMIT  .COM
 2/B:CDB      .COM  1/B:CPM60    .COM  0/B:DDT      .COM  0/B:EREMOTE .COM
 0/B:SPEEDSP  .COM  0/B:PIP      .COM  0/B:PROTOSP  .COM  0/B:RX      .COM
 0/B:TXA      .COM  0/B:EPUB     .COM  0/B:EPRIV    .COM  0/B:WSC     .COM
 0/B:X        .COM  0/B:CRCK     .COM  0/B:XSUB     .COM  0/B:DU      .COM
 0/B:QERA     .COM  0/B:FINDALL  .COM  0/B:MOVEF    .COM  0/B:REMOTE  .COM
 0/B:LOCAL    .COM  0/B:DUMP     .COM  0/B:MRESET   .COM  0/B:ELOCAL  .COM
 0/B:PUTCPMF5.COM  0/B:TEST     .COM  0/B:FDUMP    .COM  0/B:INVIS   .COM
 0/B:L80      .COM  0/B:LIST     .COM  0/B:PUB      .COM  0/B:LOAD    .COM
 0/B:MAC      .COM  0/B:SCRUB    .COM  0/B:RXA      .COM  0/B:STAT    .COM
 0/B:TX       .COM  0/B:ERASEALL.COM  0/B:WM       .COM  0/B:MSFORMAT.COM
 0/B:STATUS   .COM  0/B:UNERA    .COM  0/B:MSINIT   .COM  0/B:VIS     .COM
 0/B:WSVTIP   .COM  0/B:XD       .COM  0/B:NEWVE    .COM  0/B:DDUMP   .COM
 0/B:FORMATMA.COM  0/B:PRIV     .COM  0/B:FCOMP    .COM  0/B:DDUMPA  .COM
 0/B:PUTSYS1C.COM  0/B:DDUMPNI  .COM  0/B:DSTAT    .COM  0/B:ASM     .COM
 2/B:CDBTEST  .COM  0/B:OLDSYS   .COM  0/B:E        .COM  2/B:F/C     .COM
 3/B:ERASE    .COM  3/B:FUNKEY   .COM  3/B:DATE     .COM  3/B:FIND    .COM

Press Space Bar to continue....
 3/B:SPACE    .COM  3/B:UNERASE  .COM  3/B:MAKE     .COM  3/B:MOVE    .COM
 1/B:PUTSYSWX.COM  3/B:TIME     .COM  3/B:ASSIGN   .COM  3/B:SPEED   .COM
 3/B:PROTOCOL.COM  0/B:PRINTC   .COM  3/B:T        .COM
```

```
#define VN "1.0 02/11/83"

/* FIND - Esta utilidad puede visualizar un mapa mostrando qué discos y en
   qué números de usuario existen ficheros que coinciden con el nombre
   ambiguo especificado o bien los nombres reales que coinciden. */

#include <LIBRARY.H>

struct _dirpb dir_pb;            /* Bloque de parámetros de manejo del directorio */
struct _dir *dir_entry;          /* Apuntador a entrada al directorio (en
                                      dir_pb) */
struct _scb scb;                 /* Bloque de control de búsqueda */

char file_name[20];              /* Formateado para visualizar : un/d:Nombre-de-fichero.tipo */

short cur_disk;                  /* Disco lógico en curso al comienzo del programa */
int mcount;                      /* Cuenta de coincidencias (números de ficheros coincidentes) */
int dmcount;                     /* Cuenta de coincidencias por disco */
int lcount;                      /* Cuenta de líneas (para líneas visualizadas) */

int map_flag;                    /* 0 = mostrar nombres de fichero que coinciden.
                                      NZ = mostrar mapa de número de fichero */

        /* La matriz siguiente se utiliza para tabular los resultados para cada
           unidad de disco, y para cada número de usuario en ella.
           En suma, se han añadido dos usuarios "extra" con valores "libre"
           y "utilizado". */
```

Figura 11-5. FIND.C, un programa de utilidad que localiza ficheros o grupos de ficheros específicos.

```
unsigned disk_map[16][18];      /* Disco A -> P, usuarios 0 -> 15, libres, usados */
#define USED_COUNT 16           /* Número de usuario para entidades utilizadas */
#define FREE_COUNT 17           /* Número de usuario para entidades libres */


main(argc,argv)
short argc;             /* Cuenta de argumentos */
char *argv[];           /* Vector de argumentos (apuntador a una matriz de caracteres) */
{

printf("\nFIND Version %s (Library %s)",VN,LIBVN);
chk_use(argc);                  /* Comprobación de uso */
cur_disk = bdos(GETDISK);       /* Obtención de disco en curso por defecto */

dm_clr(disk_map);               /* Puesta a cero del mapa de disco */

        /* Fijación del bloque de control de búsqueda
           disco, nombre, tipo, número de extensión y número de bytes a
           comparar -- en este caso contrastar todos los usuarios pero únicamente
           la extensión 0 */
setscb(scb,argv[1],'?',0,13);   /* Fijación de disco, nombre y tipo */

map_flag = usstrcmp("NAMES",argv[2]);   /* Fijación de indicador para opciones de mapeado */

lcount = dmcount = mcount = 0;          /* Inicialización de contadores */

for (scb.scb_disk = 0;          /* Comienzo con disco lógico A: */
     scb.scb_disk < 16;         /* Hasta disco lógico P: */
     scb.scb_disk++)            /* Desplazamiento a disco lógico siguiente */
{

        /* Comprobación de si el disco en curso se ha seleccionado para búsqueda */
if (!(scb.scb_adisks & (1 << scb.scb_disk)))
        continue;       /* No, ignora este disco */

printf("\nSearching disk : %c",(scb.scb_disk + 'A'));
lcount++;               /* Actualización cuenta de líneas */

dir_pb.dp_disk = scb.scb_disk;  /* Fijación del disco a buscar */
dmcount = 0;                    /* Puesta a cero de contador de coincidencias */

if (!map_flag)          /* Si se han de visualizar los nombres de fichero */
        putchar('\n');  /* Movimiento hasta columna 1 */

/* Fijación del directorio a "cerrado" y forzar la función get_nde
   para abrirlo */
dir_pb.dp_open = 0;

        /* Mientras no se llegue al final del directorio, fijar apuntador
           a entrada siguiente */
while(dir_entry = get_nde(dir_pb))
        {
        /* Comprobación de entrada utilizada para actualización
           de contadores libre/utilizado */

        if (dir_entry -> de_userno == 0xE5)     /* No utilizada */
                disk_map[scb.scb_disk][FREE_COUNT]++;
        else    /* En uso */
                disk_map[scb.scb_disk][USED_COUNT]++;

        /* Seleccionar únicamente las entradas activas que están en la primera
           extensión (numeradas 0) del fichero que coincide con el nombre
           proporcionado por el usuario  */

        if (
            (dir_entry -> de_userno != 0xE5) &&
            (dir_entry -> de_extent == 0) &&
            (comp_fname(scb,dir_entry) == NAME_EQ)
           )
                {

                mcount++;       /* Actualización cuenta coincidencias */
                dmcount++;      /* Cuenta por disco */

                if (map_flag)   /* Comprobación de opción de salida */
                        {
                                /* Actualización mapa de disco */
```

Figura 11-5. (Continuación.)

```
                        disk_map[scb.scb_disk][dir_entry -> de_userno]++;
                        }
                else                    /* Visualización de nombres */
                        {
                        conv_dfname(scb.scb_disk,dir_entry,file_name);
                        printf("%s  ",file_name);

                                /* Comprobación de necesidad de comienzo de línea nueva */
                        if (!(dmcount % 4))
                                {
                                putchar('\n');
                                        if (++lcount > 18)
                                                {
                                                lcount = 0;
                                                printf("\nPress Space Bar to continue....");
                                                getchar();
                                                putchar('\n');
                                                }
                                        }
                                }
                        }
                } /* Final de directorio */
        } /* Búsqueda en todos los discos */

if (map_flag)
{
printf("\n                  Numbers show files in each user number.");
printf("\n                              --- User Numbers ---                    Dir. Entries");

dm_disp(disk_map,scb.scb_adisks);       /* Visualización mapa de disco */
}

if (mcount == 0)
printf("\n --- File Not Found --- ");

bdos(SETDISK,cur_disk); /* Puesta a disco en curso */
}


chk_use(argc)           /* Comprobación de uso */
/* Esta función comprueba que se haya especificado
   el número correcto de parámetros, avisando instrucciones
   en caso contrario.


/* Parámetros de entrada */
int argc;       /* Cuenta del número de argumentos en línea de orden */
{

/* El valor mínimo de argc es 1 (por el propio nombre del
   programa); por tanto, es siempre mayor en una unidad al número
   de parámetros de la línea de orden */

if (argc == 1 || argc > 3)
{
printf("\nUsage :");
printf("\n\tFIND d:filename.typ [NAMES]");
printf("\n\t      *:filename.typ (All disks)");
printf("\n\t      ABCD..OP:filename.typ (Selected Disks)");
printf("\n\tNAMES option shows actual names rather than map.");
exit();
}

}
```

Figura 11-5. (Continuación.)

SPACE—Muestra el espacio de disco utilizado

La utilidad SPACE que se muestra en la figura 11-6 registra los discos lógicos especificados y compone un mapa de disco que muestra, para cada

número de usuario, en cada disco lógico, cuántos Kbytes han sido utilizados. Visualiza también el número total de Kbytes utilizados y los que quedan libres en cada disco lógico.

A continuación se muestra un ejemplo de diálogo de consola que presenta a SPACE en operación:

```
P3B>space<CR>
SPACE Version 1.0 02/11/83 (Library 1.0)
Usage :
        SPACE *         (All disks)
        SPACE ABCD..OP (Selected Disks)

P3B>space *<CR>
SPACE Version 1.0 02/11/83 (Library 1.0)
Searching disk : A
Searching disk : B
Searching disk : C
                    Numbers show space used in kilobytes.
                                --- User Numbers ---              Space (Kb)
     0   1   2   3   4   5   ...   10  11  12  13  14  15  Used Free
A:  18 202      38                                          258 1196
B: 692 432 656 548  36                                     2364  996
C: 140                                                      140  204
```

```
#define VN "1.0 02/11/83"

/* SPACE -- Esta utilidad visualiza un mapa mostrando la cantidad de memoria
   (expresada en porcentajes relativos) ocupada por cada número
   de usuario para cada disco lógico. Muestra también la cantidad
   de espacio libre. */

#include <LIBRARY.H>

struct _dirpb dir_pb;           /* Bloque de parámetros de manejo del directorio */
struct _dir *dir_entry;         /* Apuntador a entrada al directorio */
struct _scb scb;                /* Bloque de control de búsqueda */
struct _dpb dpb;                /* Bloque de parámetros de disco de CP/M */

char file_name[20];             /* Formateado para visualizar: und/d:Nombre-de-fichero.tipo */

short cur_disk;          /* Disco lógico en curso al principio del programa
                            NZ = mostrar mapa de número de ficheros */
int count;               /* Utilizado para acceso a los números de bloque
                            en cada entrada del directorio */
int user;                /* Utilizado para acceso al mapa de disco cuando se calcula */

/* La matriz siguiente se utiliza para tabular los resultados para
   cada unidad de disco, y para cada número de usuario en ella.
   En suma, se han añadido dos usuarios "extra" con valores "libre"
   y "utilizado".
*/
unsigned disk_map[16][18];      /* Disco A -> P, usuarios 0 -> 15, libres, usados */
#define USED_COUNT 16           /* Número de usuario para entidades utilizadas */
#define FREE_COUNT 17           /* Número de usuario para entidades libres */


main(argc,argv)
short argc;              /* Cuenta de argumentos */
char *argv[];            /* Vector de argumentos (apuntador a una matriz de caracteres) */
{
```

Figura 11-6. SPACE.C, una utilidad que visualiza la cantidad de memoria de disco utilizada o disponible.

```
printf("\nSPACE Version %s (Library %s)",VN,LIBVN);
chk_use(argc);                  /* Comprobación de uso */
cur_disk = bdos(GETDISK);       /* Obtención de disco en curso por defecto */

dm_clr(disk_map);               /* Puesta a cero del mapa de disco */

ssetscb(scb,argv[1]);           /* Versión especial: fijación de disco,
                                   nombre y tipo */

for (scb.scb_disk = 0;          /* Comienzo con disco lógico A: */
     scb.scb_disk < 16;         /* Hasta disco lógico P: */
     scb.scb_disk++)            /* Desplazamiento a disco lógico siguiente */
        {

        /* Comprobación de si el disco en curso se ha seleccionado para búsqueda */
        if (!(scb.scb_adisks & (1 << scb.scb_disk)))
                continue;       /* No, ignorar este disco */

        printf("\nSearching disk : %c",(scb.scb_disk + 'A'));
        dir_pb.dp_disk = scb.scb_disk;  /* Fijación al disco objeto de búsqueda */

        /* Fijación del directorio a "cerrado" y forzar la función get_nde
           para abrirlo */
        dir_pb.dp_open = 0;

        /* Mientras no se llegue al final del directorio, fijar apuntador
           a entrada siguiente */
        while (dir_entry = get_nde(dir_pb))
                {
                if (dir_entry -> de_userno == 0xE5)
                        continue;       /* Ignorar entradas no activas */

                for (count = 0;         /* Comienzo por el primer bloque */
                     count < dir_pb.dp_nabpde;  /* Para número de bloques por entrada del directorio */
                     count++)
                        {
                        if (dir_pb.dp_nabpde == 8)      /* Suposición de 8 números de 2 bytes */
                                {
                                disk_map[scb.scb_disk][dir_entry -> de_userno]
                                        += (dir_entry -> _dirab.de_long[count] > 0 ? 1 : 0);
                                }
                        else    /* Suposición de 16 números de 1 byte */
                                {
                                disk_map[scb.scb_disk][dir_entry -> de_userno]
                                        += (dir_entry -> _dirab.de_short[count] > 0 ? 1 : 0);
                                }
                        }       /* Procesados todos los bloques */
                }       /* Final de directorio para este disco */


        /* Cálculo de la memoria utilizada multiplicando el número
           de bloques contados por el número de Kbytes en cada
           bloque. */

        for (user = 0;  /* Comienzo por usuario 0 */
             user < 16; /* Final con usuario 15 */
             user ++)   /* Movimiento al número de usuario siguiente */
                {
                        /* Cálculo de tamaño ocupado en Kbytes */
                disk_map[scb.scb_disk][user] *= dir_pb.dp_absize;
                        /* Realización de la suma para este disco */
                disk_map[scb.scb_disk][USED_COUNT] += disk_map[scb.scb_disk][user];
                }

        /* Espacio libre = (número de bloques * número de Kbytes por bloque)
                - Kbytes utilizados
                - Entradas del directorio * 32/1024 ... o dividido por 32 */
        disk_map[scb.scb_disk][FREE_COUNT] = (dir_pb.dp_nab * dir_pb.dp_absize)
                - disk_map[scb.scb_disk][USED_COUNT]
                - (dir_pb.dp_nument >> 5);      /* Igual que / 32 */
        }       /* Procesados todos los discos */


printf("\n                    Numbers show space used in kilobytes.");
printf("\n                              --- User Numbers ---                    Space (Kb)");

dm_disp(disk_map,scb.scb_adisks);   /* Visualización del mapa de disco */
```

Figura 11-6. (Continuación.)

```
bdos(SETDISK,cur_disk); /* Puesta a disco en curso */
}

ssetscb(scb,ldisks)     /* Versión especial de fijación de bloque de control de búsqueda */

/* Esta función activa un bloque de control de búsqueda de acuerdo con
   los discos lógicos especificados. El disco se especifica como
   una cadena simple de caracteres sin separadores. Un asterisco significa
   "todos los discos". Por ejemplo --

        ABGH    (discos A:, B:, G:, H:)
        *       (todos los discos por los que SELDSK tiene tablas)

   Fija el mapa de bits de acuerdo con los discos que deben ser objeto
   de búsqueda. Para cada disco seleccionado, comprueba si se ha generado
   algún error durante la selección del disco (es decir, si existen tablas
   para el disco en el BIOS).
   El nombre y tipo de fichero y el número de extensión se fijan a "?" para
   coincidencias con todas las entradas posibles al directorio. */

/* Parámetros de entrada */
struct _scb *scb;       /* Apuntador al bloque de control de búsqueda */
char *ldisks;           /* Apuntador a los discos lógicos */

/* Parámetros de salida
   Ninguno.
*/
{
int disk;               /* Número de disco actualmente objeto de comprobación */
unsigned adisks;        /* Mapa de bits para discos activos */

adisks = 0;             /* Suposición de que no existen discos activos */

if (*ldisks)            /* Algunos valores especificados */
        {
        if (*ldisks == '*')     /* Comprobación de si todos los discos */
                {
                adisks = 0xFFFF;        /* Fijación de todos los bits */
                }
        else                    /* Fijación de discos específicos */
                {
                while(*ldisks)  /* Hasta alcanzar final de discos */
                        {
                        /* Construcción del mapa de bits por obtención del
                           id. del disco siguiente (A - P), convirtiéndolo en un
                           número en el rango (0 - 15) y desplazando a la
                           izquierda tantos lugares como el valor 1 y haciendo
                           el OR con el disco activo en curso.
                        */
                        adisks |= 1 << (toupper(*ldisks) - 'A');
                        ++ldisks;       /* Desplazamiento a carácter siguiente */
                        }
                }
        }
else    /* Utiliza únicamente el disco por defecto en curso */
        {
        /* Fijación del bit correspondiente al disco en curso */
        adisks = 1 << bdos(GETDISK);
        }

        /* Fijación del número de usuario, nombre y tipo de fichero y extensión a "?"
           de manera que coincidirán todas las entradas activas del directorio */
                /*          0123456789012       */
strcpy(&scb -> scb_userno,"??????????????");

        /* Realización de llamadas a la rutina SELDSK del BIOS para asegurar
           que todas las unidades de disco activas tienen definidas tablas
           de disco en el BIOS. En caso contrario, puesta a cero
           de los bits correspondientes en el mapa de bits. */


for (disk = 0;          /* Comienzo con el disco A: */
     disk < 16;         /* Hasta el disco P: */
     disk++)            /* Utilización del disco siguiente */
        {
        if ( !((1 << disk) & adisks))
                continue;               /* Evitar selección de discos no especificados */
```

Figura 11-6. (Continuación.)

```
        if (biosh(SELDSK,disk) == 0)      /* Realización ce llamada a SELDSK del BIOS */
                {                         /* Devolución de 0 si disco no válido */
                /* Puesta a cero del bit correspondiente en la máscara
                   haciendo la operación AND con la máscara que tiene
                   todos los demás bits a uno. */
                adisks &= ((1 << disk) ^ 0xFFFF);
                }
        }

scb -> scb_adisks = adisks;       /* Fijación de mapa de bits en scb */

} /* Final ssetscb */


chk_use(argc)             /* Comprobación de uso */
/* Esta función comprueba que se haya especificado el número
   correcto de parámetros, avisando instrucciones en caso
   contrario. */

/* Parámetros de entrada */
int argc;         /* Cuenta del número de argumentos en línea de orden */
{

        /* El valor mínimo de argc es 1 (por el propio nombre del programa);
           por tanto, argc es siempre mayor en una unidad al número
           de parámetros de la línea de orden */

if (argc != 2)
        {
        printf("\nUsage :");
        printf("\n\tSPACE *         (All disks)");
        printf("\n\tSPACE ABCD..OP (Selected Disks)");
        exit();
        }
} /* Final chk_use */
```

Figura 11-6. (Continuación.)

MOVE—Transferencia de ficheros entre números de usuario

La utilidad MOVE que se muestra en la figura 11-7 transfiere ficheros de un número de usuario a otro en el mismo disco lógico. El movimiento se realiza cambiando el número de usuario de todas las entradas relevantes del directorio. Esto es mucho más rápido que copiar los ficheros. Evita también tener copias múltiples del mismo fichero en el disco.

Aquí se muestra un diálogo de consola que presenta a MOVE en operación:

```
P3B>move<CR>
MOVE Version 1.0 02/10/83 (Library 1.0)
Usage :
        MOVE d:filename.typ to_user {from_user} {NAMES}
             *:filename.typ (All disks)
             ABCD..OP:filename.typ (Selected Disks)
        NAMES option shows names of files moved.

P3B>dir *.com<CR>
B: ERASE   COM : FUNKEY  COM : DATE    COM : FIND    COM
B: SPACE   COM : UNERASE COM : MAKE    COM : MOVE    COM
B: TIME    COM : ASSIGN  COM : SPEED   COM : PROTOCOL COM
```

```
P3B>move *.com 0 names<CR>
MOVE Version 1.0 02/10/83 (Library 1.0)

Moving file(s)  3/B:????????.COM -> User 0.
0/B:ERASE    .COM  0/B:FUNKEY  .COM  0/B:DATE    .COM  0/B:FIND     .COM
0/B:SPACE    .COM  0/B:UNERASE .COM  0/B:MAKE    .COM  0/B:MOVE     .COM
0/B:TIME     .COM  0/B:ASSIGN  .COM  0/B:SPEED   .COM  0/B:PROTOCOL.COM

P3B>user 0<CR>
P0B>dir
B: ERASE    COM : FUNKEY   COM : DATE     COM : FIND      COM
B: SPACE    COM : UNERASE  COM : MAKE     COM : MOVE      COM
B: TIME     COM : ASSIGN   COM : SPEED    COM : PROTOCOL  COM
```

```
#define VN "1.0 02/10/83"

/* MOVE -- Esta utilidad transfiere ficheros de un número de usuario a otro
   en el mismo disco lógico. Los ficheros no se copian realmente,
   sino que se cambian sus entradas al directorio. */

#include <LIBRARY.H>

struct _dirpb dir_pb;           /* Bloque de parámetros de manejo del directorio */
struct _dir *dir_entry;         /* Apuntador a entrada del directorio */
struct _scb scb;                /* Bloque de control de búsqueda */


#define DIR_BSZ 128             /* Tamaño de la memoria intermedia de directorio */
char dir_buffer[DIR_BSZ];       /* Memoria intermedia de directorio */

char file_name[20];             /* Formateado para visualizar: un/d:Nombre-de-fichero.tipo */
short name_flag;                /* NZ para visualizar nombres de ficheros transferidos */

short cur_disk;                 /* Disco lógico en curso al principio del programa */
int from_user;                  /* Número de usuario desde el que se transfieren ficheros */
int to_user;                    /* Número de usuario al que se transfieren ficheros */

int mcount;                     /* Cuenta de coincidencias (número de ficheros coincidentes) */
int dmcount;                    /* Cuenta de coincidencias por disco */
int lcount;                     /* Cuenta de líneas (a visualizar) */


main(argc,argv)
short argc;             /* Cuenta de argumentos */
char *argv[];           /* Vector de argumentos (apuntador a matriz de caracteres) */
{

printf("\nMOVE Version %s (Library %s)",VN,LIBVN);

chk_use(argc);                  /* Comprobación de uso */

to_user = atoi(argv[2]);        /* Conversión de número de usuario a entero */
        /* Fijación y comprobación de número de usuario en destino */
if(to_user > 15)
        {
        printf("\nError -- the destination user number cannot be greater than 15.");
        }

        /* Fijación número de usuario en curso */
from_user = bdos(GETUSER,0xFF);

        /* Comprobación de especificación de número de usuario fuente */
if (isdigit(argv[3][0]))
        {
                /* Fijación y comprobación de número de usuario fuente */
        if((from_user = atoi(argv[3])) > 15)
                {
                printf("\nError -- the source user number cannot be greater than 15.");
                exit();
                }
                /* Fijación de nombre de supresión de indicador */
```

Figura 11-7. MOVE.C, un programa de utilidad que cambia números de usuario de los ficheros.

```
        name_flag = usstrcmp("NAMES",argv[4]);
        }
else                    /* Número de usuario fuente no especificado */
        {
                        /* Fijación de indicador de supresión de nombre desde parámetro */
        name_flag = usstrcmp("NAMES",argv[3]);
        }

        /* Para simplificar la lógica de lo que sigue, name_flag debe hacerse
           NZ si es igual a NAME_EQ, o 0 si es otro valor */
name_flag = (name_flag == NAME_EQ ? 1 : 0);

if (to_user == from_user)           /* A = desde */
        {
        printf("\nError - 'to' user number is the same as 'from' user number.");
        exit();
        }

        /* Fijación del bloque de control de búsqueda, nombre de fichero,
           tipo, número de usuario, nombre y tipo. Como el número de extensión
           no forma parte de la comparación, se encontrarán todas
           las extensiones de un fichero dado. */
setscb(scb,argv[1],from_user,'?',13);

cur_disk = bdos(GETDISK);           /* Obtención del disco en curso por defecto */
lcount = dmcount = mcount = 0;      /* Inicialización de contadores */

for (scb.scb_disk = 0;              /* Comienzo con disco lógico A: */
     scb.scb_disk < 16;             /* Hasta el disco lógico P: */
     scb.scb_disk++)                /* Desplazamiento a siguiente disco lógico */
        {
                /* Comprobación de que se ha seleccionado para búsqueda el disco en curso */
        if (!(scb.scb_adisks & (1 << scb.scb_disk)))
                continue;           /* No, ignorar el disco */
                /* Conversión del nombre y número de usuario de búsqueda para salida */
        conv_dfname(scb.scb_disk,scb,file_name);
        printf("\n\nMoving file(s) %s -> User %d.",file_name,to_user);

        lcount++;                   /* Actualización contador de líneas */

        dir_pb.dp_disk = scb.scb_disk;  /* Fijación al disco objeto de búsqueda */
        dmcount = 0;                    /* Puesta a cero del contador de coincidencias del disco */

        if (name_flag)              /* Si se han de visualizar los nombres de fichero */
                putchar('\n');      /* Desplazamiento a columna 1 */

                /* Fijación del directorio a "cerrado" y forzar la función get_nde
                   para abrirlo */
        dir_pb.dp_open = 0;

                /* Mientras no se llegue al final del directorio, fijar apuntador
                   a entrada siguiente */
        while(dir_entry = get_nde(dir_pb))
                {
                        /* Dar como coincidentes todas las entradas que tengan el número
                           de usuario, nombre y tipo de fichero correctos y un número
                           de extensión cualquiera. */
                if (
                    (dir_entry -> de_userno != 0xE5) &&
                    (comp_fname(scb,dir_entry) == NAME_EQ)
                   )
                        {

                        dir_entry -> de_userno = to_user;       /* Movimiento a nuevo usuario */
                                /* Petición de escritura de sector anterior */
                        dir_pb.dp_write = 1;

                        mcount++;       /* Actualización de cuenta de coincidencias */
                        dmcount++;      /* Cuenta por disco */

                        if (name_flag)  /* Comprobación de opción de salida */
                                {
                                conv_dfname(scb.scb_disk,dir_entry,file_name);
                                        printf("%s  ",file_name);

                                /* Comprobación de necesidad de comienzo de nueva línea */
```

Figura 11-7. (Continuación.)

```
                                        if (!(dmcount % 4))
                                                {
                                                putchar('\n');
                                                if (++lcount > 18)
                                                {
                                                lcount = 0;
                                                printf("\nPress Space Bar to continue....");
                                                getchar();
                                                putchar('\n');
                                                }
                                        }
                                }
                        }
                }
        }

if (mcount == 0)
        printf("\n --- No Files Moved --- ");

bdos(SETDISK,cur_disk);  /* Puesta a disco en curso */
}


chk_use(argc)              /* Comprobación de uso */
/* Esta función comprueba que se haya especificado
   el número correcto de parámetros, avisando instrucciones
   en caso contrario. */

/* Parámetros de entrada */
int argc;         /* Cuenta del número de argumentos en línea de orden */
{

/* El valor mínimo de argc es 1 (por el propio nombre del programa);
   por tanto, argc es siempre mayor en una unidad al número
   de parámetros de la línea de orden. */

if (argc == 1 || argc > 5)
        {
        printf("\nUsage :");
        printf("\n\tMOVE d:filename.typ to_user {from_user} {NAMES}");
        printf("\n\t     *:filename.typ (All disks)");
        printf("\n\t     ABCD..OP:filename.typ (Selected Disks)");
        printf("\n\tNAMES option shows names of files moved.");
        exit();
        }

}
```

Figura 11-7. (Continuación.)

Otras utilidades

Los programas de utilidad descritos en esta sección no son en absoluto un conjunto completo. Puede desearse desarrollar muchos otros programas útiles especializados. Algunas posibilidades son:

FILECOPY
: Una versión más especializada de PIP podría copiar grupos de ficheros especificados ambiguamente. De especial importancia sería la capacidad de leer un fichero que contenga los nombres de los ficheros que se han de copiar. Una opción útil sería la posibilidad de detectar el valor del bit de atributo de fichero no utilizado y copiar únicamente ficheros en que ha sido cambiado.

PROTECT/UNPROTECT
: Este par de utilidades permitiría "esconder" ficheros en números de

usuario mayores que 15. Los ficheros así escondidos no serían accesibles más que por medio de un procesamiento de UNPROTECT, y al mismo tiempo trasladándolos de nuevo al rango normal de números de usuario.

RECLAIM

Esta utilidad leería todos los sectores de un disco (utilizando el BIOS). Todos los sectores dañados que se encuentren podrán ser suprimidos lógicamente mediante la creación de una entrada en el directorio de ficheros, con los números de bloques que "reservarían" efectivamente los bloques que contuvieran los sectores dañados.

OWNER

Dado un número de pista o de sector, esta utilidad accederá al directorio y determinará cuál o cuáles ficheros están utilizando esa parte del disco. Esto es útil si se tiene un sector o pista dañados en un disco. Entonces se puede determinar cuáles son los ficheros que se han dañado.

Programas de utilidad para el BIOS mejorado

Esta sección describe algunos programas de utilidad que trabajan con el BIOS mejorado que se muestra en la figura 8-10. Algunas de estas utilidades trabajan directamente con los dispositivos físicos del sistema, que pueden variar de una computadora a otra. La cabecera de biblioteca contiene #declaraciones de definición para números de dispositivo y nombres para dispositivos físicos (figuras 11-2, f y 11-2, g).

Estas #sentencias de definición se utilizan para construir una tabla de códigos de dispositivos físicos. Si se tienen más dispositivos físicos o se desea cambiar los nombres con los que se hace referencia a los mismos, se necesitará cambiar estas definiciones.

Todas estas utilidades comparten algunas características en la forma en que son invocadas. Si se llaman sin parámetros, visualizan instrucciones en la consola relativas a la clase de parámetros disponibles. Si se llaman con la palabra "SHOW" (o "S", "SH", etc.) como parámetro, visualizan la situación actual de cualquier atributo de la utilidad.

MAKE—Conversión de ficheros en "invisibles" o "visibles"

La utilidad MAKE que se muestra en la figura 11-8 está diseñada para operar en conjunción con la opción de ficheros públicos implementados en el BIOS mejorado de la figura 8-10. Tiene dos formas de operación —conversión de ficheros en "invisibles" o "visibles".

Un fichero invisible es uno del usuario 0 que ha sido puesto en modo de sólo lectura y sistema. Cuando la opción de ficheros públicos está disponible, estos ficheros no pueden verse cuando se utiliza la orden DIR, ni pueden ser borrados accidentalmente.

Un fichero visible es el que se ha fijado en modo de lectura/escritura y de directorio.

Cuando los ficheros se han hecho invisibles, se transfieren desde el número de usuario actual al usuario 0. Cuando se hacen visibles, son transferidos desde el usuario 0 al número de usuario en curso.

A continuación damos un ejemplo de diálogo de consola que muestra a MAKE en operación:

```
P3B>make<CR>
MAKE Version 1.0 02/12/83 (Library 1.0)

Usage :
        MAKE d:filename.typ INVISIBLE (NAMES)
                            VISIBLE
             *:filename.typ (All disks)
             ABCD..OP:filename.typ (Selected Disks)
        NAMES option shows names of files processed.

P3B>dir *.com<CR>
B: ERASE    COM : UNERASE  COM : ASSIGN   COM : PROTOCOL COM

P3B>make *.com invisible names<CR>
MAKE Version 1.0 02/12/83 (Library 1.0)

Moving files from User 3 to 0 and making them Invisible.
Searching disk : B

          0/B:ERASE   .COM made Invisible in User 0.
          0/B:UNERASE .COM made Invisible in User 0.
          0/B:ASSIGN  .COM made Invisible in User 0.
          0/B:PROTOCOL.COM made Invisible in User 0.

P3B>make erase.com visible names<CR>
MAKE Version 1.0 02/12/83 (Library 1.0)

Moving files from User 0 to 3 and making them Visible.
Searching disk : B

          3/B:ERASE   .COM made Visible in User 3.
```

```
#define VN "1.0 02/12/83"

/* MAKE - Esta utilidad contiene en realidad dos programas similares;
   dependiendo cada uno de un parámetro especificado en la
   línea de orden.

   INVISIBLE encuentra todos los ficheros especificados, los transfiere
   al usuario 0 y los pone en modo sistema y en estado de
   sólo lectura. Estos ficheros pueden ser utilizados por números de usuario
   distintos de cero si se activa la prestación de ficheros públicos
   del BIOS.
```

Figura 11-8. MAKE.C, una utilidad que convierte los ficheros en "invisibles" y protegidos o los hace "visibles", accesibles y no protegidos.

```
    VISIBLE es el opuesto al anterior. Los ficheros especificados se
    transfieren al número de usuario en curso y cambiado a modo
    directorio y estado de lectura/escritura. */

#include <LIBRARY.H>

struct _dirpb dir_pb;           /* Bloque de parámetros de manejo del directorio */
struct _dir *dir_entry;         /* Apuntador a entrada del directorio */
struct _scb scb;                /* Bloque de control de búsqueda */
short to_user;                  /* Número de usuario al que deben ponerse los ficheros */
short from_user;                /* Número de usuario desde el que se transferirán los ficheros */

char file_name[20];             /* Formateado para visualización: un/d:Nombre-de-fichero.tipo */
short name_flag;                /* NZ para visualización de los nombres de fichero transferidos */

short cur_disk;                 /* Disco lógico en curso al comienzo del programa */

int mcount;                     /* Cuenta de coincidencias (número de nombres coincidentes) */

short invisible;                /* NZ cuando el parámetro especifica invisible */
char *operation;                /* Apuntador a "visible" o "invisible" */

main(argc,argv)
short argc;             /* Cuenta de argumentos */
char *argv[];           /* Vector de argumentos (apuntador a matriz de caracteres) */
{

printf("\nMAKE Version %s (Library %s)",VN,LIBVN);
chk_use(argc);                  /* Comprobación de uso */
cur_disk = bdos(GETDISK);       /* Obtención de disco en curso por defecto */
mcount = 0;                     /* Inicialización contador */


        /* Fijación del indicador de invisible de acuerdo con el parámetro */
invisible = usstrcmp("VISIBLE",argv[2]);

        /* Fijación de los números from_user y to_user (usuario desde y hacia) dependiendo
           del programa construido y de los parámetros especificados. */

if (invisible)
        {
        from_user = bdos(GETUSER,0xFF); /* Obtención número de usuario en curso */
        to_user = 0;    /* Transferencia de ficheros siempre a usuario 0 */
        operation = "Invisible";        /* Fijación de apuntador a cadena */
        }
else    /* visible */
        {
        from_user = 0;                  /* Transferencia siempre desde usuario 0 */
        to_user = bdos(GETUSER,0xFF);   /* Obtención de usuario en curso */
        operation = "Visible";          /* Fijación de apuntador a cadena */
        }

        /* Fijación del bloque de control de búsqueda, disco, nombre, tipo, número
           de usuario, de extensión y número de bytes a comparar -- en este caso,
           se comparan todas las extensiones del usuario -- desde. */
setscb(scb,argv[1],from_user,'?',13);   /* Fijación de disco, nombre y tipo */

name_flag = usstrcmp("NAMES",argv[3]);  /* Fijación indicador de supresión de nombre según parámetro 3 */

        /* Para simplificar la lógica de lo que sigue, name_flag debe hacerse
           NZ si es igual a NAME_EQ, o 0 si es otro valor */
name_flag = (name_flag == NAME_EQ ? 1 : 0);


        /* Conversión de nombre y número de usuario para salida */
conv_dfname(scb.scb_disk,scb,file_name);
printf("\n\nMoving files from User %d to %d and making them %s.",
        from_user,to_user,operation);

for (scb.scb_disk = 0;          /* Comienzo con disco lógico A: */
     scb.scb_disk < 16;         /* Hasta el disco lógico P: */
     scb.scb_disk++)            /* Desplazamiento a siguiente disco lógico */
        {

                /* Comprobación de que se ha seleccionado para búsqueda el disco en curso */
        if (!(scb.scb_adisks & (1 << scb.scb_disk)))
                continue;       /* No, ignorar este disco */
```

Figura 11-8. (Continuación.)

```
        printf("\nSearching disk : %c",(scb.scb_disk + 'A'));

        dir_pb.dp_disk = scb.scb_disk;   /* Fijar disco objeto de búsqueda */

        if (name_flag)              /* Si se han de visualizar los nombres de fichero */
                putchar('\n');      /* Desplazamiento a columna 1 */


                /* Fijación del directorio a "cerrado" y forzar la función get_nde
                   para abrirlo. */
        dir_pb.dp_open = 0;

                /* Mientras no se llegue al final del directorio, fijar apuntador
                   a entrada siguiente. */
        while(dir_entry = get_nde(dir_pb))
                {

                        /* Dar como coincidentes todas las entradas que tengan el número
                           de usuario, nombre y tipo de fichero correctos y un número
                           de extensión cualquiera. */
                if (
                    (dir_entry -> de_userno != 0xE5) &&
                    (comp_fname(scb,dir_entry) == NAME_EQ)
                   )
                        {
                        mcount++;           /* Actualización cuentas de coincidencia */

                        if (invisible)
                                {           /* Fijación bits ms */
                                dir_entry -> de_fname[8] |= 0x80;
                                dir_entry -> de_fname[9] |= 0x80;
                                }
                        else    /* Visible */
                                {           /* Borrado de los bits ms */
                                dir_entry -> de_fname[8] &= 0x7F;
                                dir_entry -> de_fname[9] &= 0x7F;
                                }

                                /* Transferencia a número de usuario correcto */
                        dir_entry -> de_userno = to_user;

                                /* Indicación de escritura de sector anterior */
                        dir_pb.dp_write = 1;

                                /* Comprobación de visualización de nombre */
                        if (name_flag)
                                {
                                conv_dfname(scb.scb_disk,dir_entry,file_name);
                                printf("\n\t%s made %s in User %d.",
                                        file_name,operation,to_user);
                                }
                        }
                }       /* Procesadas todas las entradas al directorio */
        }       /* Procesados todos los discos */

if (mcount == 0)
        printf("\n --- No Files Processed --- ");

bdos(SETDISK,cur_disk);  /* Puesta a disco en curso */
}


chk_use(argc)               /* Comprobación de uso */
/* Esta función comprueba que se haya especificado
   el número correcto de parámetros, avisando instrucciones
   en caso contrario. */
*/
/* Parámetros de entrada */
int argc;         /* Cuenta del número de argumentos en línea de orden */
{

         /* El valor mínimo de argc es 1 (por el propio nombre del programa);
            por tanto, argc es siempre mayor en una unidad al número
            de parámetros de la línea de orden. */

if (argc == 3 || argc == 4)
        return;
```

Figura 11-8. (Continuación.)

```
else
        {
        printf("\nUsage :");
        printf("\n\tMAKE d:filename.typ INVISIBLE (NAMES)");
        printf("\n\t                    VISIBLE");
        printf("\n\t     *:filename.typ (All disks)");
        printf("\n\t     ABCD..OP:filename.typ (Selected Disks)");
        printf("\n\tNAMES option shows names of files processed.");
        exit();
        }

}
```

Figura 11-8. (Continuación.)

SPEED—Fijación de relación de velocidad (baud-rate)

La utilidad SPEED que se muestra en la figura 11-9 fija el valor de la velocidad de transmisión para un dispositivo serie específico. A continuación se presenta un ejemplo de diálogo de consola que muestra alguna de las opciones:

```
P3B>speed<CR>
SPEED 1.0 02/17/83
The SPEED utility sets the baud rate speed for each physical device.
Usage is :  SPEED physical-device baud-rate, or
            SPEED SHOW      (to show current settings)
Valid physical devices are:
                TERMINAL
                PRINTER
                MODEM
Valid baud rates are:
                300
                600
                1200
                2400
                4800
                9600
                19200

P3B>speed show<CR>
SPEED 1.0 02/17/83
Current Baud Rate settings are :
        TERMINAL set to 9600 baud.
        PRINTER set to 9600 baud.
        MODEM set to 9600 baud.

P3B>speed m 19<CR>
SPEED 1.0 02/17/83
Current Baud Rate settings are :
        TERMINAL set to 9600 baud.
        PRINTER set to 9600 baud.
        MODEM set to 19200 baud.

P3B>speed xyz 12<CR>
SPEED 1.0 02/17/83
Physical Device 'XYZ' is invalid or ambiguous.
Legal Physical Devices are :
                TERMINAL
                PRINTER
                MODEM
```

```
#define VN "\nSPEED 1.0 02/17/83"

/* Esta utilidad fija la relación de velocidad de transmisión para
   cada dispositivo físico. */

#include <LIBRARY.H>

struct _ct ct_pdev[MAXPDEV + 2];          /* Tabla de dispositivo físico */

        /* Items de hardware específico */

                                          /* Relación de velocidad para puertos serie (baud rate) */
#define B300    0x35                      /* 300 baudios */
#define B600    0x36                      /* 600 baudios */
#define B1200   0x37                      /* 1200 baudios */
#define B2400   0x3A                      /* 2400 baudios */
#define B4800   0x3C                      /* 4800 baudios */
#define B9600   0x3E                      /* 9600 baudios */
#define B19200  0x3F                      /* 19200 baudios */
struct _ct ct_br[10];     /* Tabla de códigos para relación de baudios (entrada de repuestos) */


        /* Parámetros en línea de orden */
#define PDEV argv[1]      /* Dispositivo físico */
#define BAUD argv[2]      /* Relación de baudios (baud rate) */


main(argc,argv)
int argc;
char *argv[];
{
printf(VN);       /* Visualización de mensaje de "funcionando" */
setup();          /* Activar tabla de códigos */
chk_use(argc);    /* Comprobación de uso correcto */

        /* Comprobación de existencia de petición de mostrar valores en curso */
if (usstrcmp("SHOW",argv[1]))
        {                 /* No, suposición de petición de fijación */
        set_baud(get_pdev(PDEV),get_baud(BAUD));  /* Fijar velocidad de transmisión */
        }

show_baud();              /* Visualización de valores en curso */

} /* final de programa */

setup()                   /* Fijación de tabla de dispositivo físico */
{
        /* Inicialización tabla de dispositivo físico */
ct_init(ct_pdev[0],T_DEVN,PN_T);          /* Terminal */
ct_init(ct_pdev[1],P_DEVN,PN_P);          /* Impresora */
ct_init(ct_pdev[2],M_DEVN,PN_M);          /* Modem */
ct_init(ct_pdev[3],CT_SNF,"*");   /* Finalizador */

        /* Inicialización tabla de velocidad de transmisión */
ct_init(ct_br[0],B300,"300");
ct_init(ct_br[1],B600,"600");
ct_init(ct_br[2],B1200,"1200");
ct_init(ct_br[3],B2400,"2400");
ct_init(ct_br[4],B4800,"4800");
ct_init(ct_br[5],B9600,"9600");
ct_init(ct_br[6],B19200,"19200");
ct_init(ct_br[7],CT_SNF,"*");     /* Finalizador */
}

unsigned
get_pdev(ppdev)           /* Obtención de dispositivo físico */
/* Esta función devuelve el código de dispositivo físico especificado
   por el usuario en la línea de orden. */
char *ppdev;              /* Apuntador a cadena de caracteres */
{
unsigned retval;                          /* Devolución de valor */

retval = ct_parc(ct_pdev,ppdev);          /* Obtención de código para cadena ASCII */
if (retval == CT_SNF)             /* Si no se encuentra la cadena */
```

Figura 11-9. SPEED.C, una utilidad que fija la velocidad de transmisión para un dispositivo específico.

```
        {
        printf("\n\007Physical Device '%s' is invalid or ambiguous.",
                ppdev);
        printf("\nLegal Physical Devices are : ");
        ct_disps(ct_pdev);          /* Visualización de todos los valores */
        exit();
        }
return retval;                      /* Devolución de código */
}

unsigned
get_baud(pbaud)
/* Esta función devuelve la constante de tiempo de la velocidad de transmisión
   para la velocidad especificada por el usuario en la línea de órdenes */
char *pbaud;                /* Apuntador a cadena de caracteres */
{
unsigned retval;                        /* Devolución de valor */
retval = ct_parc(ct_br,pbaud);  /* Obtención de código para cadena ASCII */
if (retval == CT_SNF)           /* Si no se encuentra la cadena */
        {
        printf("\n\007Baud Rate '%s' is invalid or ambiguous.",
                pbaud);
        printf("\nLegal Baud Rates are : ");
        ct_disps(ct_br);            /* Visualización de todos los valores */
        exit();
        }
return retval;          /* Devolución de código */
}

set_baud(pdevc,baudc)   /* Fijación de la velocidad del dispositivo especifico */
int pdevc;              /* Código del dispositivo físico */
short baudc;            /* Código de la velocidad de transmisión */
                        /* En algunos sistemas éste es un
                           valor de dos bytes (sin signo) */
{
short *baud_rc;         /* Apuntador a la constante de relación de velocidad */
                        /* En algunos sistemas éste es un
                           valor de dos bytes (sin signo) */
/* Nota: Los códigos respectivos para acceder a las constantes de relación
   de velocidad via la función get_cba son:
        Dispositivo #0 = 19, #2 = 21, #2 = 23. Esta función utiliza
   esta relación matemática */

        /* Fijación de apuntador a constante de relación de velocidad */
baud_rc = get_cba(CB_DO_BRC + (pdevc << 1));

        /* Fijación de constante de relación de velocidad */
*baud_rc = baudc;

        /* Llamada a la rutina de inicialización del BIOS */
bios(CIOINIT,pdevc);
}

show_baud()             /* Mostrar velocidad de transmisión en curso */
{

int pdevn;              /* Número de dispositivo físico */
short baudc;            /* Código de velocidad de transmisión */
                        /* En algunos sistemas éste es un
                           valor de dos bytes (sin signo) */
short *baud_rc;         /* Apuntador a la constante de relación de velocidad */
                        /* En algunos sistemas éste es un
                           valor de dos bytes (sin signo) */
/* Nota: Los códigos respectivos para acceder a las constantes de relación
   de velocidad via la función get_cba son:
        Dispositivo #0 = 19, #1 = 21, #2 = 23. Esta función utiliza
   esta relación matemática  */

printf("\nCurrent baud rate settings are :");


for (pdevn = 0; pdevn <= MAXPDEV; pdevn ++)      /* Todos los dispositivos físicos */
        {
                /* Fijación de apuntador a constante de relación de velocidad --
                   el código para la función get_cba se calcula añadiendo
                   el número de dispositivo físico *2 al código de relación
                   de velocidad para el dispositivo #0 */
```

Figura 11-9. (Continuación.)

```
        baud_rc = get_cba(CB_DO_BRC + (pdevn << 1));

                /* Fijación de constante de relación de velocidad */
        baudc = *baud_rc;

        printf("\n\t%s set to %s baud.",
                ct_strc(ct_pdev,pdevn),  /* Obtención de apuntador a nombre de dispositivo */
                ct_strc(ct_br,baudc) );  /* Obtención de apuntador a relación de velocidad */
        }
}

chk_use(argc)           /* Comprobación de uso correcto */
int argc;               /* Cuenta de argumentos */
{

if (argc == 1)
        {
        printf("\nThe SPEED utility sets the baud rate speed for each physical device.");
        printf("\nUsage is :  SPEED physical-device baud rate, or");
        printf("\n            SPEED SHOW     (to show current settings)");
        printf("\n\nValid physical devices are: ");
        ct_disps(ct_pdev);
        printf("\nValid baud rates are: ");
        ct_disps(ct_br);
        exit();
        }
}
```

Figura 11-9. (Continuación.)

PROTOCOL—Fijación de protocolos de línea serie

La utilidad PROTOCOL que se muestra en la figura 11-10 se utiliza para fijar el protocolo para un dispositivo serie específico.

Los controladores de cada dispositivo físico pueden soportar varios protocolos de línea serie. Los protocolos se dividen en dos grupos, dependiendo de si se aplican a la salida de datos o a la entrada de la computadora.

Obsérvese que los protocolos de salida DTR y de entrada RTS pueden coexistir con otros protocolos. La estrategia es, en primer lugar, fijar el protocolo basado en carácter requerido y entonces fijar el protocolo DTR/RTS. Un ejemplo de esto se muestra en el siguiente diálogo de consola:

```
P3B>protocol<CR>
PROTOCOL Vn 1.0 02/17/83
PROTOCOL sets the physical device's serial protocols.
        PROTOCOL physical-device direction protocol {message-length}

Legal physical devices are :
                TERMINAL
                PRINTER
                MODEM

Legal direction/protocols are :
                Output DTR
                Output XON
                Output ETX
                Input RTS
                Input XON

        Message length can be specifed with Output ETX.
```

```
P3B>protocol show<CR>
PROTOCOL Vn 1.0 02/17/83
        Protocol for TERMINAL - None.
        Protocol for PRINTER - Output XON
        Protocol for MODEM - Input RTS

P3B>protocol m o e 128<CR>
PROTOCOL Vn 1.0 02/17/83
        Protocol for TERMINAL - None.
        Protocol for PRINTER - Output XON
        Protocol for MODEM - Output ETX  Message Length 128 bytes.

P3B>protocol m o d<CR>
PROTOCOL Vn 1.0 02/17/83
        Protocol for TERMINAL - None.
        Protocol for PRINTER - Output XON
        Protocol for MODEM - Output DTR Output ETX  Message Length
          128 bytes.
```

```
#define VN "\nPROTOCOL Vn 1.0 02/17/83"
/* PROTOCOL -- Esta utilidad fija el protocolo de puerto serie para el dispositivo
   físico especificado. Como alternativa visualiza los protocolos
   en curso para todos los dispositivos serie. */

#include <LIBRARY.H>

        /* Tablas de códigos utilizados para relacionar cadenas ASCII con códigos */
struct _ct ct_iproto[3];          /* Tabla de código para protocolos de entrada */
struct _ct ct_oproto[4];          /* Tabla de códigos para protocolos de salida */
struct _ct ct_dproto[7];          /* Tabla de códigos para protocolos de visualización */
struct _ct ct_pdev[MAXPDEV + 2]; /* Tabla de dispositivo físico */
struct _ct ct_io[3];              /* Entrada, salida */


        /* Parámetros en la línea de orden */
#define PDEV argv[1]      /* Dispositivo físico */
#define IO argv[2]        /* Entrada, salida */
#define PROTO argv[3]     /* Protocolo */
#define PROTOL argv[4]    /* Longitud del mensaje del protocolo */

main(argc,argv)
int argc;
char *argv[];
{
printf(VN);       /* Visualización del mensaje de "funcionando" */
setup();          /* Activar tablas de códigos */
chk_use(argc);    /* Comprobación de uso correcto */

        /* Comprobación de petición de mostrar valores en curso */
if (usstrcmp("SHOW",argv[1]))
        {                 /* No, se supone que es necesaria fijación */
        set_proto(get_pdev(PDEV),        /* Dispositivo físico */
                          /* Entrada/salida y protocolo */
                get_proto(get_io(IO),PROTO),
                PROTOL);          /* Longitud de mensaje de protocolo */
        }
show_proto();

} /* final de programa */

setup()                   /* Fijación de las tablas de códigos para este programa */
{
        /* Inicialización de la tabla de dispositivo físico */
ct_init(ct_pdev[0],0,PN_T);       /* Terminal */
ct_init(ct_pdev[1],1,PN_P);       /* Impresora */
ct_init(ct_pdev[2],2,PN_M);       /* Modem */
ct_init(ct_pdev[3],CT_SNF,"*"); /* Finalizador */
```

Figura 11-10. PROTOCOL.C, una utilidad que fija el protocolo que gobierna la entrada y la salida de un dispositivo serie específico.

```
        /* Inicialización de tabla de entrada/salida */
ct_init(ct_io[0],0,"INPUT");
ct_init(ct_io[1],1,"OUTPUT");
ct_init(ct_io[2],CT_SNF,"*");              /* Finalizador */

        /* Inicialización de tabla de protocolo de salida */
ct_init(ct_oproto[0],DT_ODTR,"DTR");
ct_init(ct_oproto[1],DT_OXON,"XON");
ct_init(ct_oproto[2],DT_OETX,"ETX");
ct_init(ct_oproto[3],CT_SNF,"*");          /* Finalizador */

        /* Inicialización de tabla de protocolo de entrada */
ct_init(ct_iproto[0],DT_IRTS,"RTS");
ct_init(ct_iproto[1],DT_IXON,"XON");
ct_init(ct_iproto[2],CT_SNF,"*");          /* Finalizador */

        /* Inicialización de protocolo de visualización */
ct_init(ct_dproto[0],DT_ODTR,"Output DTR");
ct_init(ct_dproto[1],DT_OXON,"Output XON");
ct_init(ct_dproto[2],DT_OETX,"Output ETX");
ct_init(ct_dproto[3],DT_IRTS,"Input RTS");
ct_init(ct_dproto[4],DT_IXON,"Input XON");
ct_init(ct_dproto[5],CT_SNF,"*");
}

unsigned
get_pdev(ppdev)            /* Obtención de dispositivo físico */
/* Esta función devuelve el código de dispositivo físico
   especificado por el usuario en la línea de orden. */
char *ppdev;               /* Apuntador a cadena de caracteres */
{
unsigned retval;           /* Devolución de valor */

retval = ct_parc(ct_pdev,ppdev); /* Obtención de código para cadena ASCII */
if (retval == CT_SNF)            /* Si no se encuentra la cadena */
        {
        printf("\n\007Physical Device '%s' is invalid or ambiguous.",
                ppdev);
        printf("\nLegal Physical Devices are : ");
        ct_disps(ct_pdev);       /* Visualización de todos los valores */
        exit();
        }
return retval;                   /* Devolución de código */
}

unsigned
get_io(pio)                /* Obtención de parámetro de entrada/salida */
char *pio;                 /* Apuntador a cadena de caracteres */
{
unsigned retval;                 /* Devolución de valor */

retval = ct_parc(ct_io,pio);     /* Obtención de código para cadena ASCII */
if (retval == CT_SNF)            /* Si no se encuentra la cadena */
        {
        printf("\n\007Input/Output direction '%s' is invalid or ambiguous.",
                pio);
        printf("\nLegal values are : ");
        ct_disps(ct_io);         /* Visualización de todos los valores */
        exit();
        }
return retval;                   /* Devolución de código */
}

unsigned

get_proto(output,pproto)
/* Esta función devuelve el código del protocolo especificado
   por el usuario en la línea de orden. */
int output;                /* =1 para salida, =0 para entrada */
char *pproto;              /* Apuntador a cadena de caracteres */

{
unsigned retval;                 /* Devolución de valor */

if (output)                      /* Salida especificada */
        {
```

Figura 11-10. (Continuación.)

```
                /* Obtención de código para cadena ASCII */
        retval = ct_parc(ct_oproto,pproto);
        if (retval == CT_SNF)           /* Si no se encuentra la cadena */
                {
                printf("\n\007Output Protocol '%s' is invalid or ambiguous.",
        pproto);
                printf("\nLegal Output Protocols are : ");
                ct_disps(ct_oproto);    /* Visualización todos los protocolos válidos */
                exit();
                }
        }
else                                    /* Entrada especificada */
        {
                /* Obtención de código para cadena ASCII */
        retval = ct_parc(ct_iproto,pproto);
        if (retval == CT_SNF)           /* Si no se encuentra la cadena */
                {
                printf("\n\007Input Protocol '%s' is invalid or ambiguous.",
        pproto);
                printf("\nLegal Input Protocols are : ");
                ct_disps(ct_iproto);    /* Visualización todos los protocolos válidos */
                exit();
                }
        }
return retval;                          /* Devolución de código */
}


set_proto(pdevc,protoc,pplength) /* Fijación de protocolo para dispositivo físico */
int pdevc;                              /* Código de dispositivo físico */
unsigned protoc;                        /* Byte de protocolo */
char *pplength;                         /* Apuntador a longitud de protocolo */
{
struct _ppdt
{
char *pdt[16];          /* Matriz de 16 apuntadores a tablas de dispositivos */
} ;
struct _ppdt *ppdt;                     /* Apuntador a la matriz de tablas de dispositivo */
struct _dt *dt;                         /* Apuntador a tabla de dispositivo */

ppdt = get_cba(CB_DTA); /* Fijación de apuntador a matriz de apuntadores */
dt = ppdt -> pdt[pdevc];

if (!dt)                        /* Comprobación de validez del apuntador a la matriz */
        {
        printf("\nError -- Array of Device Table Addresses is not set for device #%d.",
                pdevc);
        exit();
        }

if (protoc & 0x8000)        /* Comprobación de si el byte de protocolo se fija
                               directamente o debe ser operado (OR) */
        {                       /* OR */
        dt -> dt_st1 |= (protoc & 0x7F);
        }
else
        {                       /* Fijado directamente */
        dt -> dt_st1 = (protoc & 0x7F);
        }

if ((protoc & 0x7F) == DT_OETX) /* Si ETX/ACK, comprobación de longitud
                                   de mensaje */
        {
        if (isdigit(*pplength))         /* Comprobación de longitud presente */
                {
                        /* Conversión de la longitud a binario y fijación del campo
                           de tabla de dispositivo. */
                dt -> dt_etxml = atoi(pplength);
                }
        }
}


show_proto()                    /* Mostrar valores de protocolo en curso */
{
struct _ppdt
{
```

Figura 11-10. (Continuación.)

```
char *pdt[16];          /* Matriz de 16 apuntadores a tablas de dispositivos */
} ;
struct _ppdt *ppdt;             /* Apuntador a la matriz de tablas de dispositivo */
struct _dt *dt;                 /* Apuntador a tabla de dispositivo */
int pdevc;                      /* Código de dispositivo físico */
struct _ct *dproto;             /* Apuntador a protocolo de visualización */

ppdt = get_cba(CB_DTA);  /* Fijación de apuntador a matriz de apuntador */

        /* Para todos los dispositivos físicos */

for (pdevc = 0; pdevc <= MAXPDEV; pdevc++)
        {
                /* Fijación de apuntador a tabla de dispositivos */
        dt = ppdt -> pdt[pdevc];

        if (dt) /* Comprobación de validez de apuntador a matriz */
                {
                printf("\n\tProtocol for %s - ",ct_strc(ct_pdev,pdevc));
                        /* Check if any protocols set */
                if (!(dt -> dt_st1 & ALLPROTO))
                        {
                        printf("None.");
                        continue;
                        }

                        /* Fijación de apuntador a tabla de protocolos de visualización */
                dproto = ct_dproto;
                while (dproto -> _ct_code != CT_SNF)
                        {
                                /* Comprobación de fijado bit de protocolo */
                        if (dproto -> _ct_code & dt -> dt_st1)
                                {       /* Visualización de protocolo */
                                printf("%s ",dproto -> _ct_sp);
                                }
                        ++dproto;       /* Desplazamiento a entrada siguiente */
                        }
                        /* Comprobación de protocolo ETX/ACK y
                           longitud de mensaje a visualizar */
                if (dt -> dt_st1 & DT_OETX)
                        printf(" Message length %d bytes.",
                                dt -> dt_etxml);
                }
        }
}

chk_use(argc)           /* Comprobación de uso correcto */
int argc;               /* Cuenta de argumentos en línea de orden */
{
if (argc == 1)
        {
        printf("\nPROTOCOL sets the physical device's serial protocols.");
        printf("\n\tPROTOCOL physical-device direction protocol [message-length]");
        printf("\n\nLegal physical devices are :");
        ct_disps(ct_pdev);
        printf("\nLegal direction/protocols are :");
        ct_disps(ct_dproto);
        printf("\n\tMessage length can be specifed with Output ETX.\n");
        exit();
        }
}
```

Figura 11-10. (Continuación.)

ASSIGN—Asignación de dispositivos físicos a lógicos

La utilidad ASSIGN mostrada en la figura 11-11 pone a uno los bits necesarios de los bits de redireccionamiento de entrada/salida física del BIOS. Asigna la entrada/salida de un dispositivo lógico a un dispositivo

físico. La entrada sólo puede darse desde un solo dispositivo físico, mientras que la salida puede ser dirigida a múltiples dispositivos.

He aquí un ejemplo de diálogo de consola que muestra a ASSIGN en acción:

```
P3B>assign<CR>
ASSIGN Vn 1.0 02/17/83
ASSIGN sets the Input/Output redirection.
        ASSIGN logical-device INPUT physical-device
        ASSIGN logical-device OUTPUT physical-dev1 {phy_dev2..}
        ASSIGN SHOW     (to show current assignments)

Legal logical devices are :
                CONSOLE
                AUXILIARY
                LIST

Legal physical devices are :
                TERMINAL
                PRINTER
                MODEM

P3B>assign show<CR>
ASSIGN Vn 1.0 02/17/83
Current Device Assignments are :
        CONSOLE INPUT is assigned to -  TERMINAL
        CONSOLE OUTPUT is assigned to -  TERMINAL
        AUXILIARY INPUT is assigned to -  MODEM
        AUXILIARY OUTPUT is assigned to -  MODEM
        LIST INPUT is assigned to -  PRINTER
        LIST OUTPUT is assigned to -  PRINTER

P3B>assign a o t m p<CR>
ASSIGN Vn 1.0 02/17/83
Current Device Assignments are :
        CONSOLE INPUT is assigned to -  TERMINAL
        CONSOLE OUTPUT is assigned to -  TERMINAL
        AUXILIARY INPUT is assigned to -  MODEM
        AUXILIARY OUTPUT is assigned to -  TERMINAL PRINTER MODEM
        LIST INPUT is assigned to -  PRINTER
        LIST OUTPUT is assigned to -  PRINTER
```

```
#define VN "\nASSIGN Vn 1.0 02/17/83"

#include <LIBRARY.H>

struct _ct ct_pdev[MAXPDEV + 2];        /* Tabla de dispositivo físico */


        /* Nombres de dispositivo lógico */
#define LN_C    "CONSOLE"
#define LN_A    "AUXILIARY"
#define LN_L    "LIST"
struct _ct ct_ldev[4];          /* Tabla de dispositivo lógico */

struct _ct ct_io[3];            /* Entrada/salida */

        /* Parámetros en línea de orden */
#define LDEV argv[1]    /* Dispositivo lógico */
#define IO argv[2]      /* Entrada/salida */
```

Figura 11-11. ASSIGN.C, una utilidad que asigna la entrada y salida de un dispositivo lógico a dos dispositivos físicos.

```
main(argc,argv)
int argc;
char *argv[];
{

printf(VN);       /* Visualización de mensaje de funcionando */
setup();          /* Activar tablas de códigos */
chk_use(argc);    /* Comprobación de uso correcto */

        /* Comprobación de petición de mostrar valores en curso */
if (usstrcmp("SHOW",argv[1]))
        {               /* No, suposición de fijado necesario */

                /* NOTA: El número de dispositivos físicos a tratar
                   viene dado por argc - 3 */
        set_assign(get_ldev(LDEV),get_io(IO),argc - 3,argv);
        }
show_assign();

}

setup()                 /* Fijación de tablas de código para este programa */
{
        /* Inicialización de la tabla de dispositivos físicos */
ct_init(ct_pdev[0],0,PN_T);       /* Terminal */
ct_init(ct_pdev[1],1,PN_P);       /* Impresora */
ct_init(ct_pdev[2],2,PN_M);       /* Modem */
ct_init(ct_pdev[3],CT_SNF,"*");   /* Finalizador */

        /* Inicialización tabla de dispositivos lógicos */
ct_init(ct_ldev[0],0,LN_C);       /* Terminal */
ct_init(ct_ldev[1],1,LN_A);       /* Auxiliar */
ct_init(ct_ldev[2],2,LN_L);       /* Listado */
ct_init(ct_ldev[3],CT_SNF,"*");   /* Finalizador */

        /* Inicialización tabla de entrada/salida */
ct_init(ct_io[0],0,"INPUT");
ct_init(ct_io[1],1,"OUTPUT");
ct_init(ct_io[2],CT_SNF,"*");             /* Finalizador */

}

unsigned
get_ldev(pldev)           /* Obtención dispositivo lógico */
/* Esta función devuelve el código del dispositivo lógico
   especificado por el usuario en la línea de orden. */
char *pldev;              /* Apuntador a cadena de caracteres */
{
unsigned retval;                          /* Devolución de valor */
retval = ct_parc(ct_ldev,pldev);          /* Obtención de código para cadena ASCII */
if (retval == CT_SNF)             /* Si no se encuentra la cadena */
        {
        printf("\n\007Logical device '%s' is invalid or ambiguous.",
                pldev);
        printf("\nLegal logical devices are : ");
        ct_disps(ct_ldev);        /* Visualización de todos los valores */
        exit();
        }
return retval;                    /* Devolución de código */
}

unsigned
get_io(pio)               /* Obtención de parámetro de entrada/salida */
char *pio;                /* Apuntador a cadena de caracteres */
{
unsigned retval;                   /* Devolución de valor */

retval = ct_parc(ct_io,pio);       /* Obtención de código para cadena ASCII */
if (retval == CT_SNF)              /* Si no se encuentra la cadena */
        {
        printf("\n\007Input/output direction '%s' is invalid or ambiguous.",
                pio);
        printf("\nLegal values are : ");
        ct_disps(ct_io);           /* Visualización de todos los valores */
        exit();
        }
```

Figura 11-11. (Continuación.)

```
return retval;                    /* Devolución de código */
}

set_assign(ldevc,output,argc,argv)      /* Fijar asignación (redireccionamiento de E/S) */
int ldevc;                        /* Código de dispositivo lógico */
int output;                       /* Código de redireccionamiento de E/S */
int argc;                         /* Cuenta de argumentos a tratar */
char *argv[];                     /* Réplica de parámetros para función principal */
{
unsigned *redir;                  /* Apuntador a palabra de redireccionamiento */
int pdevc;                        /* Código de dispositivo físico */
unsigned rd_val;                  /* Valor de redireccionamiento */

        /* Obtención de dirección de palabra de redireccionamiento de E/S */

            Este programa supone que los valores del código get_cba
            están ordenados:
                 Dispositivo #0, entrada y salida
                 Dispositivo #1, entrada y salida
                 Dispositivo #2, entrada y salida


            El código get_cba se calcula multiplicando el código
            del dispositivo lógico por 2 (desplazamiento de 1 bit a la
            izquierda) y añadiéndolo al código para dispositivo #0
            entrada. Finalmente se añade la variable de salida (0 = entrada,
            1 = salida)   */
redir = get_cba(CB_CI + (ldevc << 1) + output);

rd_val = 0;       /* Inicialización de valor de redireccionamiento */

        /* Para salida la asignación puede hacerse a varios dispositivos físicos,
           de manera que este programa pueda efectuarse varias veces  */
do
        {
                /* Obtención de código para cadena ASCII */
                /* NOTA: Los parámetros de dispositivo físico empiezan
                   con el parámetro #3. Sin embargo, argc es un contador
                   decrementador del número de dispositivos físicos
                   tratados. Por tanto, argc + 2 causa que sean procesados
                   en orden inverso (es decir, de derecha a izquierda
                   en la línea de orden) */

        pdevc = ct_parc(ct_pdev,argv[argc + 2]);

        if (pdevc == CT_SNF)              /* Si cadena no encontrada */
                {
                printf("\n\007Physical device '%s' is invalid or ambiguous.",
                argv[argc + 2]);
                printf("\nLegal physical devices are : ");
                ct_disps(ct_pdev);        /* Visualización de todos los valores */
                exit();
                }
                /* Repetición del bucle tantas veces como parámetros haya
                   (únicamente para salida) */
        else
                {
                /* Construcción de nuevo valor de redireccionamiento realizando la operación OR con el bit a
                   uno desplazado a la izquierda tantos lugares como pdevc. */
                rd_val |= (1 << pdevc);
                }
        } while (--argc && output);

*redir = rd_val;          /* Fijación de valor en bloque de configuración */
}

show_assign()                     /* Mostrar velocidad de transmisión en curso */
{
int rd_code;                      /* Código de redireccionamiento para get_cba */
int ldevn;                        /* Número de dispositivo lógico */
int pdevn;                        /* Número de dispositivo físico */
unsigned rd_val;                  /* Valor de redireccionamiento */
unsigned *prd_val;                /* Apuntador a valor de redireccionamiento */

/* Nota: Los códigos respectivos para acceso a los valores de redireccionamiento vía
   la función get_cba (obtención de la dirección del bloque de configuración) son:
        Dispositivo #0 entrada de consola -- 5
```

Figura 11-11. (Continuación.)

```
        Dispositivo #0 salida de consola -- 6
        Dispositivo #1 entrada auxiliar -- 7
        Dispositivo #1 salida auxiliar -- 8
        Dispositivo #2 entrada de listado -- 9
        Dispositivo #2 salida de listado -- 10

   Esta función utiliza esta relación matemática  */

printf("\nCurrent device assignments are :");

        /* Para todos los códigos get_cba */
for (rd_code = CB_CI; rd_code <= CB_LO; rd_code++)
        {
                /* Fijación de apuntador a valor de redireccionamiento */
        prd_val = get_cba(rd_code);
                /* Obtención de valor de redireccionamiento de entrada */
        rd_val = *prd_val;      /* Realiza también la inversión de byte */

                /* Visualización de nombre de dispositivo. El código rd_code
                   se convierte en número de dispositivo restando del primer
                   número de código y dividiendo por dos (desplazamiento un lugar
                   a la derecha). La dirección de entrada/salida se calcula
                   a partir del bit menos significativo de rd_code. */

        printf("\n\t%s %s is assigned to - ",
                ct_strc(ct_ldev,(rd_code - CB_CI) >> 1),
                ct_strc(ct_io,((rd_code & 0x01) ^ 1)));

                /* Para todos los dispositivos físicos */
        for (pdevn = 0; pdevn < 16; pdevn++)
                {
                        /* Comprobación de asignación del dispositivo físico en curso haciendo la
                           AND con un bit a uno desplazado a la izquierda pdevn veces */
                if (rd_val & (1 << pdevn))      /* ¿Está activado el dispositivo? */
                        {       /* Visualización nombre dispositivo físico */
                        printf(" %s",ct_strc(ct_pdev,pdevn) );
                        }
                }

        }
}


chk_use(argc)                   /* Comprobación de uso correcto */
int argc;                       /* Cuenta de argumentos en línea de orden */
{
if (argc == 1)
        {
        printf("\nASSIGN sets the Input/Output redirection.");
        printf("\n\tASSIGN logical-device INPUT physical-device");
        printf("\n\tASSIGN logical-device OUTPUT physical-dev1 [phy_dev2..]");
        printf("\n\tASSIGN SHOW    (to show current assignments)");
        printf("\n\nLegal logical devices are :");
        ct_disps(ct_ldev);
        printf("\nLegal physical devices are :");
        ct_disps(ct_pdev);
        exit();
        }
}
```

Figura 11-11. (Continuación.)

DATE—Fijación de la fecha del sistema

La utilidad DATE que se muestra en la figura 11-12 fija la fecha del sistema en el bloque de configuración, junto con un indicador que indica que la utilidad DATE ha sido utilizada. Otros programas de utilidad pueden usar a ésta como comprobación primitiva de si la fecha del sistema es actual.

A continuación se muestra un ejemplo de diálogo de consola:

```
P3B>date<CR>
DATE Vn 1.0 02/18/83
DATE sets the system date. Usage is :
        DATE mm/dd/yy
        DATE SHOW (to display current date)

P3B>date show<CR>
DATE Vn 1.0 02/18/83
        Current Date is 12/18/82

P3B>date 2/23/83<CR>
DATE Vn 1.0 02/18/83
        Current Date is 02/23/83
```

```
#define VN "\nDATE Vn 1.0 02/18/83"

 /* Esta utilidad acepta la fecha en curso como parámetro de orden,
    la valida y fija la fecha interna del sistema en el BIOS.
    Alternativamente puede ser requerida para visualizar la fecha
    en curso del sistema. */

#include <LIBRARY.H>

char *date;                 /* Apuntador a la fecha en el bloque de configuración */
char *date_flag;            /* Apuntador a indicador de fecha-fijada */
int mm,dd,yy;               /* Variables que contienen mes, día y año */
int mcount;                 /* Cuenta de contraste de los valores numéricos entrados */
int count;                  /* Cuenta utilizada para añadir ceros a la izquierda en fecha */

main(argc,argv)
int argc;
char *argv[];
{
printf(VN);                 /* Visualización de mensaje de "funcionando" */
date = get_cba(CB_DATE);            /* Fijación de apuntador a fecha */
date_flag = get_cba(CB_DTFLAGS); /* Fijación de apuntador a indicador de fecha fijada */

if (argc != 2)              /* Comprobación de petición de ayuda (help) (o necesaria) */
        show_use();         /* Visualización de utilización correcta y salida */

if (usstrcmp("SHOW",argv[1]))    /* Comprobación de no opción de mostrar */
        {
                /* Conversión de fecha en mes, día, año */
        mcount = sscanf(argv[1],"%d/%d/%d",&mm,&dd,&yy);
        if (mcount != 3)                    /* Entrada no numérica */
                show_use();         /* Visualización de utilización correcta y salida */

                /* NOTA: La comprobación de validez que sigue es simplista,
                   pero puede ser ampliada para realizar una comprobación
                   más sensible al contexto, días en el mes,
                   años bisiestos, etc. */
        if (mm > 12 || mm < 1)   /* Comprobación de mes, día y año válidos */
                {
                printf("\nMonth = %d is illegal.",mm);
                show_use();         /* Visualización de utilización correcta y salida */
                }
        if (dd > 31 || dd < 1)
                {
                printf("\nDay = %d is illegal.",dd);
                show_use();         /* Visualización de utilización correcta y salida */
                }
        if (yy > 90 || yy < 83)  /* <=== NOTA ! */
                {
                printf("\nYear = %d is illegal.",yy);
                show_use();         /* Visualización de utilización correcta y salida */
                }
```

Figura 11-12. DATE.C, una utilidad que hace que la fecha actual forme parte del sistema.

```
                /* Conversión de los enteros anteriores en una cadena formateada */
        sprintf(date,"%2d/%2d/%2d",mm,dd,yy);
        date[8] = 0x0A;             /* Terminación con salto de línea */
        date[9] = '\0';             /* Finalizador de nueva cadena */

                /* Cambio de " 1/ 2/ 3" en "01/02/03" */
        for (count = 0; count < 7; count+=3)
                {
                if (date[count] == ' ')
                        date[count] = '0';
                }

                /* Puesta a uno del indicador que indica que ha sido puesta la fecha */
        *date_flag |= DATE_SET;
        }
printf("\n\tCurrent Date is %s",date);
}

show_use()                  /* Visualización de uso correcto y salida */
{
printf("\nDATE sets the system date. Usage is :");
printf("\n\tDATE mm/dd/yy");
printf("\n\tDATE SHOW (to display current date)\n");
exit();
}
```

Figura 11-12. (Continuación.)

TIME—Fijación de la hora del sistema

La utilidad TIME que se muestra en la figura 11-13 fija la hora del sistema. Igual que DATE, TIME fija un indicador para que otras utilidades puedan comprobar si la hora del sistema está actualizada.

He aquí un ejemplo de diálogo de consola:

```
P3B>time<CR>
TIME Vn 1.0 02/18/83
TIME sets the system time. Usage is :
        TIME hh[:mm[:ss]]
        TIME SHOW (to display current time)

P3B>time show<CR>
TIME Vn 1.0 02/18/83
        Current Time is 13:08:44

P3B>time 5:47<CR>
TIME Vn 1.0 02/18/83
        Current Time is 05:47:00
```

```
#define VN "\nTIME Vn 1.0 02/18/83"

/* Esta utilidad acepta la hora como parámetro de orden, la valida
   y fija la hora interna del sistema en el BIOS.
   Alternativamente visualiza la hora del sistema. */

#include <LIBRARY.H>

char *time;                 /* Apuntador a hora en el bloque de configuración */
char *time_set;             /* Apuntador a indicador de hora fijada */
int hh,mm,ss;               /* Variables que contienen horas, minutos, segundos */
int mcount;                 /* Cuenta para contrastar los valores numéricos entrados */
int count;                  /* Cuenta utilizada para añadir ceros a la izquierda de la hora */
```

Figura 11-13. TIME.C, una utilidad que hace que la hora actual forme parte del sistema.

```
main(argc,argv)
int argc;
char *argv[];
{
printf(VN);                 /* Visualización de mensaje de "funcionando" */
time = get_cba(CB_TIMEA);       /* Fijación de apuntador a hora */
time_flag = get_cba(CB_DTFLAGS);        /* Fijación de apuntador a indicador
                                           de hora fijada */
hh = mm = ss = 0;           /* Inicialización de la hora si no se han especificado
                               los minutos y los segundos */

if (argc != 2)              /* Comprobación de petición de ayuda (help) (o necesaria) */
        show_use();         /* Visualización de uso correcto y salida */

if (usstrcmp("SHOW",argv[1]))   /* Comprobación de no opción SHOW */
        {
                /* Conversión de la hora en horas, minutos y segundos */
        mcount = sscanf(argv[1],"%d:%d:%d",&hh,&mm,&ss);
        if (!mcount)            /* Entrada no numérica */
                show_use();     /* Visualización de uso correcto y salida */

        if (hh > 12)            /* Comprobación de horas, minutos y segundos válidos */
                {
                printf("\n\007Hours = %d is illegal.",hh);
                show_use();     /* Visualización de uso correcto y salida */
                }
        if (mm > 59)
                {
                printf("\n\007Minutes = %d is illegal.",mm);
                show_use();     /* Visualización de utilidad correcta y salida */
                }
        if (ss > 59)
                {
                show_use();     /* Visualización de utilidad correcta y salida */
                printf("\n\007Seconds = %d is illegal.",ss);
                }

                /* Conversión de los enteros anteriores en una cadena formateada */
        sprintf(time,"%2d:%2d:%2d",hh,mm,ss);
        time[8] = 0x0A;         /* Terminación con salto de linea */
        time[9] = '\0';         /* Finalizador de nueva cadena */

                /* Cambio de " 1: 2: 3" en "01:02:03" */
        for (count = 0; count < 7; count+=3)
                {
                if (time[count] == ' ')
                        time[count] = '0';
                }
               /* Puesta a uno del indicador que indica que ha sido puesta la fecha */
        *time_flag |= TIME_SET;
        }

printf("\n\tCurrent Time is %s",time);
}

show_use()                  /* Visualización de uso correcto y salida */
{
printf("\nTIME sets the system time. Usage is :");
printf("\n\tTIME hh[:mm[:ss]]");
printf("\n\tTIME SHOW (to display current time)\n");
exit();
}
```

Figura 11-13. (Continuación.)

FUNKEY—Fijación de las teclas de función

La utilidad FUNKEY que se muestra en la figura 11-14 fija las cadenas de caracteres asociadas con teclas de función específicas. En la cadena de

caracteres especificados, el carácter "<" se convierte en un carácter LINE FEED (salto de línea). He aquí un ejemplo de diálogo de consola:

```
P3B>funkey<CR>
FUNKEY sets a specific function key string.
        FUNKEY key-number "string to be programmed<"
                    (Note : '<' is changed to line feed.)
                    (        key-number is from 0 to 17.)
                    (        string can be up to 16 chars.)
        FUNKEY SHOW         (displays settings for all keys)

P3B>funkey show<CR>
FUNKEY Vn 1.0 02/18/83
        Key #0 = 'Function Key 1<'
        Key #1 = 'Function Key 2<'

P3B>funkey 0 "PIP B:=A:*.*[V]<"<CR>

P3B>funkey show<CR>
FUNKEY Vn 1.0 02/18/83
        Key #0 = 'PIP B:=A:*.*[V]<'
        Key #1 = 'Function Key 2<'
```

```
#define VN "\nFUNKEY Vn 1.0 02/18/83"

#include <LIBRARY.H>


int fnum;                           /* Número de tecla función a programar */
char fstring[20];                   /* Cadena para tecla-función */
struct _fkt *pfk;                   /* Apuntador a tabla de teclas-función */

main(argc,argv)
int argc;
char *argv[];
{

if (argc == 1 || argc > 3)
    {  show_use();

pfk = get_cba(CB_FKT);  /* Fijación de apuntador a tabla de teclas-función */

if (usstrcmp("SHOW",argv[1]))
        {
        if (!isdigit(argv[1][0]))
                {
                printf("\n\007'%s' is an illegal function key.",
                        argv[1]);
                show_use();
                }

        fnum = atoi(argv[1]);   /* Conversión de número de tecla-función */

        if (fnum > FK_ENTRIES)
                {
                printf("\n\007Function key number %d too large.",fnum);
                show_use();
                }

        if (get_fs(fstring) > FK_LENGTH)
                {
                printf("\n\007Function key string is too long.");
                show_use();
                }
```

Figura 11-14. FUNKEY.C, una utilidad que fija las cadenas de caracteres asociadas con teclas de función específicas.

```
        pfk += fnum;    /* Actualización de apuntador a cadena */
                /* Copia de la cadena en la tabla de teclas-función */

                /* Comprobación de tecla-función presente */
        if (!(pfk -> fk_input[0]))
                {
                printf("\n\007Error : Function Key #%d is not set up to be programmed.",fnum);
                show_use();
                }
        strcpy(pfk -> fk_output,fstring);
        }
else            /* Función SHOW especificada */
        {
        printf(VN);             /* Visualización de mensaje de "funcionando" */
        show_fun();
        }
}

get_fs(string)          /* Obtención de cadena de función de parámetros de orden */
char string[];          /* Apuntador a cadena de caracteres */
{
char *tail;             /* Apuntador a parámetros de orden */
short tcount;           /* Cuenta de caracteres totales en parámetros de orden */
int slen;               /* Longitud de la cadena */

tail = 0x80;            /* La línea de orden está en la posición 0080H de la memoria */
tcount = *tail++;       /* Fijación del número total de caracteres en la zona de parámetros de orden */
slen = 0;               /* Inicialización de longitud de cadena */

while(tcount--)         /* Para todos los caracteres en zona de parámetros de orden */
        {
        if (*tail++ == '"')     /* Examen para primeras acotaciones */
                break;
        }
if (!tcount)            /* No se han encontrado acotaciones */
        {
        printf("\n\007No leading quotes found.");
        show_use();
        }

++tcount;               /* Ajuste del contador de caracteres */
while(tcount--)         /* Para resto de caracteres en zona de parámetros de orden */
        {
        if (*tail == '"')
                {
                string[slen] = '\0';    /* Añadir finalizador */
                break;          /* Salida del bucle */
                }
        string[slen] = *tail++; /* Transferencia de carácter en zona de parámetros a cadena */

        if (string[slen] == '<')
                string[slen] = 0x0A;
        ++slen;
        }
if (!tcount)            /* No encontradas marcas de final */
        {
        printf("\n\007No trailing quotes found.");
        show_use();
        }
return slen;            /* Devolución de longitud de cadena */
}


show_fun()              /* Visualización de valores para todas las teclas-función */
{
struct _fkt *pfkt;      /* Apuntador local a teclas-función */
int count;              /* Cuenta para acceso a tecla-función */
char *lf;               /* Apuntador a carácter "<" (salto de línea) */

pfkt = get_cba(CB_FKT); /* Fijación de apuntador a tabla de teclas-función */
for (count = 0; count <= FK_ENTRIES; count++)
        {
        if (pfkt -> fk_input[0])        /* Tecla programada */
                {
                /* Comprobación de final físico de tabla */
```

Figura 11-14. (Continuación.)

```
                if (pfkt -> fk_input == 0xFF)
                        break;   /* Si salto fuera del bucle */
                strcpy(fstring,pfkt -> fk_output);
                        /* Conversión de todos los caracteres 0x0A a "<" */
                while (lf = strscn(fstring,"\012"))
                        {
                        *lf = '<';
                        }

                printf("\n\tKey #%d = '%s'",count,fstring);
                }
        ++pfkt;         /* Desplazamiento a entrada siguiente */
        }
}


show_use()
{
printf("\nFUNKEY sets a specific function key string.");
printf("\n\tFUNKEY key-number \042string to be programmed<\042 ");
printf("\n\t            (Note : '<' is changed to line feed.)");
printf("\n\t            (        key-number is from 0 to %d.)",
        FK_ENTRIES-1);
printf("\n\t            (        string can be up to %d chars.)",
        FK_LENGTH);
printf("\n\tFUNKEY SHOW        (displays settings for all keys)");
exit();
}
```

Figura 11-14. (Continuación.)

Otras utilidades

A causa de las limitaciones de espacio, no se pueden mostrar en este capítulo todos los posibles programas de utilidad para el BIOS. Otros que pueden necesitarse desarrollar para tener un conjunto completo son:

PUBLIC/PRIVATE
: Este par de servicios conectarán o desconectarán el indicador de ficheros públicos, haciendo los ficheros del usuario 0 asequibles a otros números de usuario, o lo contrario.

SETTERM
: Este programa programará la tabla de escape CONOUT, fijando las distintas secuencias de escape según se requieran. También puede programar los caracteres en la tabla de teclas de función que se comparan con las emitidas por el terminal en uso.

SAVESYS
: Esta utilidad salvará la configuración en curso en el bloque de configuración a largo plazo.

LOADSYS
: Este cargará el bloque de configuración a largo plazo desde una imagen previamente salvada.

DO

Esta utilidad copiará la zona de parámetros de orden en una memoria intermedia multiorden, cambiando "\" en LINE FEED y fijando entonces el apuntador de entrada forzada en la memoria intermedia multiorden. Como resultado de ello, los caracteres de la memoria intermedia multiorden se introducirán en el flujo de entrada de la consola como si hubieran sido mecanografiados de una en una orden.

SPARE

Esta utilidad trabajará de acuerdo con el manejo de sectores dañados de disco rígido en el controlador de disco. Tomará nota de los sectores y pistas dañados en el disco rígido. Hecho esto, las llamadas siguientes a estos sectores o pistas serán redirigidas a una parte distinta del disco.

Visualización de mensajes
de error
Errores varios

12

Mensajes de error

Este capítulo relaciona los mensajes de error que emanan de CP/M estándar y de sus programas de utilidad. No incluyen mensajes de error del BIOS; estos mensajes, si existen, son producto individualizado de programadores que escribieron las distintas versiones del BIOS.

Los mensajes de error se muestran en orden alfabético, seguidos (entre paréntesis) por el nombre del programa o el componente de CP/M que envía el mensaje. Los mensajes se muestran en mayúsculas, aunque el mensaje real que se vea contenga letras minúsculas. Los caracteres adicionales que se componen para "embellecer" el mensaje se han omitido. Por ejemplo, el mensaje "**ABORTED**" será relacionado como "ABORTED".

A continuación de cada mensaje hay una aclaración y, cuando es posible, alguna información para ayudar a luchar contra el error.

La última sección del capítulo se enfrenta con errores conocidos o peculiaridades del CP/M y de sus utilidades. Léase esta sección de modo que se reconozcan estos problemas cuando se presenten.

Visualización de mensajes de error

? (CCP)

El CCP compone un signo interrogativo si se introduce un nombre de orden y no existe el fichero correspondiente de "orden.COM" en el disco.

También se compone si se omite el número de páginas requerido como parámetro en la orden SAVE.

? (DDT)

El DDT envía un signo interrogativo bajo ciertas circunstancias. Debe utilizarse el contexto (y algún trabajo de adivinanza) para determinar qué es lo equivocado. Aquí hay varias causas específicas de problemas:

- El DDT no puede encontrar el fichero que se le ha pedido que cargue en memoria. Sálgase del DDT e investíguese utilizando DIR o STAT (el fichero debe estar fijado en modo sistema y por consiguiente es invisible con DIR).
- Existe un problema con los datos en el fichero HEX que se ha encargado cargar al DDT. El problema podría ser un control de error de suma (check-sum) en una línea dada o un campo no válido en cualquier parte del registro. Inténtese escribir el archivo HEX en la consola o utilizar un editor para examinarlo. Es raro encontrar sólo uno o dos bits o bytes defectuosos en un fichero HEX; posiblemente se hayan estropeado cantidades mayores de fichero. Por consiguiente, hay que ser capaz de distinguir el problema de forma correcta y rápida. Si se tiene el código fuente para el programa, hay que volver a ensamblarlo para producir otra copia del fichero HEX. Si no se tiene el código fuente, no hay un camino viable para resolver este problema a menos que se esté preparado para crear a mano el fichero HEX —una tarea difícil y pesada.
- El DDT no reconoce la instrucción que se ha introducido cuando se utiliza la orden "A" (ensamblaje) para convertir una instrucción en código fuente en hexadecimal. Compruébese la línea que se introdujo. El DDT no quiere tabuladores en la línea (aunque parezca aceptarlos) ni números hexadecimales seguidos de "H". Compruébese que los nemónicos y los operandos son válidos.

??=(DDT)

Esta notación críptica se utiliza por el DDT cuando se usa la orden "L" (desensamblado de lista) para visualizar alguna parte de memoria en el

lenguaje ensamblador primitivo del DDT. El DDT no puede traducir los 256 posibles valores de un byte. Algunos de ellos no se utilizan en el conjunto de instrucciones del 8080. Cuando el DDT encuentra un valor intraducible, compone este mensaje como código de instrucción, seguido del valor real del byte en hexadecimal.

Esto se verá si se trata de desensamblar programa escrito para la CPU Z80, que ocurre porque utiliza instrucciones 8080 no asignadas. También se verá si se intenta desensamblar bytes que contengan cadenas de texto ASCII más que instrucciones 8080.

ABORTED (STAT)

Si se introduce un carácter del teclado mientras está trabajando STAT en el directorio de ficheros fijándolos a $DIR (directorio), a $SYS (sistema), a $R/W (lectura/escritura) o a $R/O (sólo lectura), entonces compondrá este mensaje, detendrá lo que esté haciendo y ejecutará un arranque caliente.

En contraste, si se introduce la orden

```
A>stat *.*<cr>
```

para visualizar todos los ficheros de un disco, no hay forma de poder abortar el proceso.

ABORTED (PIP)

Este mensaje se forma si se pulsa cualquier carácter del teclado mientras PIP está copiando un fichero en el dispositivo de listado.

BAD DELIMITER (STAT)

Si el BIOS utiliza el método IOBYTE normal de asignar dispositivos físicos a lógicos, se utiliza STAT para realizar la asignación. La orden tiene este formato:

```
STAT RDR:=PTR:
```

STAT visualiza este mensaje si no puede encontrar el "=" en el sitio correcto.

BAD LOAD (CCP)

Este es probablemente el mensaje de error más oscuro de CP/M. Se obtendrá este mensaje si se intenta cargar un fichero COM mayor que el

área de programa transitorio. La única solución es construir un sistema CP/M que tenga un TPA mayor.

BAD PARAMETER (PIP)

PIP acepta ciertos parámetros entre corchetes al final de la línea de orden. Se visualiza este mensaje si se introduce un parámetro no válido o un valor numérico ilegal a continuación de una letra de parámetro.

BDOS ERROR ON d: BAD SECTOR (BDOS)

El BDOS compone este mensaje si las funciones READ y WRITE del BIOS vuelven siempre indicando un error. La única respuesta correcta a este mensaje es mecanografiar CONTROL-C. El CP/M ejecutará entonces un arranque caliente. Si se mecanografía CARRIAGE RETURN, el error será ignorado —con resultados impredecibles.

Un BIOS bien implementado debería incluir la recuperación y el control de errores de disco, de forma que el error no fuera comunicado nunca al BDOS. Si el BIOS proporciona la opción de ignorar un error, hágase esto únicamente cuando se esté razonablemente seguro del resultado o se tengan copias adecuadas, de forma que se puedan volver a crear los ficheros.

BDOS ERROR ON d: FILE R/O (BDOS)

Se verá este mensaje si se intenta borrar (ERA) un fichero que haya sido puesto en modo de sólo lectura. Mecanografiando cualquier carácter en el teclado, el BDOS realiza una operación de arranque caliente. Obsérvese que el BDOS no indica *cuál* es el fichero que crea el problema. Esto puede ser un problema cuando se utilizan nombres de fichero ambiguos en la orden ERA. Utilícese la orden STAT para visualizar todos los ficheros en el disco; éste comunicará cuáles están en modo de sólo lectura.

Este mensaje se forma también si un programa intenta borrar un fichero de sólo lectura. De nuevo será difícil determinar cuál de los ficheros causa el problema. La única solución es utilizar STAT para intentar inferir cuál de los ficheros de sólo lectura puede estar causándolo.

BDOS ERROR ON d: R/O (BDOS)

Este parece similar al mensaje anterior, pero se refiere a un disco lógico completo en lugar de a un fichero de sólo lectura. Sin embargo, se da

raramente porque se haya declarado un disco de sólo lectura. Normalmente, ocurre porque se cambiaron disquetes sin mecanografiar un CONTROL-C; CP/M detectará el nuevo disquete y, sin ninguna indicación exterior, lo fijará en estado de sólo lectura.

Si el operador o el programa intentan escribir cualquier dato en el disco, el intento será detectado por el BDOS y se visualizará este mensaje. El mecanografiado de cualquier carácter en el teclado provoca un arranque caliente —y entonces puede actuarse.

BDOS ERROR ON d: SELECT (BDOS)

El BDOS visualiza este mensaje si el operador o un programa intentan seleccionar un disco lógico para el cual el BIOS carece de las tablas necesarias. El BDOS utiliza el valor devuelto por SELDSK para determinar si un disco lógico "existe" o no.

Si se estuvo intentando cambiar el disco por defecto a uno no existente, se habrá de pulsar el botón RESET en la computadora. No hay forma de salir de este error.

Sin embargo, si se estuvo intentando ejecutar una orden que accedía a un disco no existente, entonces se puede mecanografiar un CONTROL-C y CP/M realizará un arranque caliente.

BREAK x AT y (ED)

Este es otro mensaje críptico cuyo significado es imposible de adivinar. La lista que sigue explica los posibles valores de "x". El valor "y" se refiere a la orden que ED estaba ejecutando cuando ocurrió el error.

x	Significado
#	Fallo de búsqueda. El ED no encuentra la cadena que se le pidió que buscase.
?	Orden no reconocida.
0	Fichero no encontrado.
>	Memoria intermedia interna del ED llena.
E	Orden abortada.
F	Disco o directorio lleno. Habrá que determinar lo que está causando el problema.

CANNOT CLOSE, READ/ONLY? (SUBMIT)

SUBMIT compone este mensaje si el disco en el que se está intentando escribir su fichero de salida, "$$$,SUB", está protegido físicamente contra la escritura. No hay que confundir esto con que el disco que está protegido *lógicamente* contra la escritura.

La versión estándar de SUBMIT escribe el fichero de salida en el disco por defecto en curso, de forma que si es otro que la unidad A:, se puede tener la posibilidad de evitar este problema si se cambia a A:, el disco por defecto, y entonces se introduce una orden de la forma

```
A>submit b:subfile<cr>
```

CANNOT CLOSE DESTINATION FILE (PIP)

PIP compone este mensaje si el disco de destino está protegido físicamente contra la escritura. Compruébese el disco de destino. Si está protegido contra la escritura, quítese la protección y repítase la operación.

Si el disco no está protegido, existe un problema de hardware. Los datos del directorio escritos en el disco se están escribiendo en un sitio erróneo, incluso en un disco equivocado o no se está grabando.

CANNOT CLOSE FILES (ASM)

ASM compone este mensaje si no puede cerrar sus ficheros de salida porque el disco está físicamente protegido contra la escritura, o si existe un problema de hardware que evita que los datos se escriban en el disco. Véase el párrafo anterior.

CANNOT READ (PIP)

PIP compone este mensaje si se intenta leer información de un dispositivo lógico que sólo puede enviar. Por ejemplo:

```
A>pip diskfile=LST:<cr>
```

PIP compondrá también este mensaje si se le confunde lo suficiente, como, por ejemplo, con las instrucciones siguientes:

```
A>pip file1=file2;file3<cr>
```

CANNOT WRITE (PIP)

PIP compone este mensaje si se intenta enviar (escribir) información a un dispositivo lógico que sólo puede ser utilizado para entrada, tal como el RDR: (lectora, el nombre anacrónico del dispositivo de entrada auxiliar).

CHECKSUM ERROR (LOAD)

LOAD compone este mensaje si encuentra una línea en el fichero de entrada HEX que no tenga el control de suma (check-sum) correcto para los datos de la línea.

LOAD compone también información que ayuda a descubrir el problema:

```
CHECKSUM ERROR
LOAD ADDRESS  0110  <- First address on line in file
ERROR ADDRESS 0112  <- Address of next byte to be loaded
BYTES READ:
0110:
0110: 00 33 22 2B 02 21 27 02    <- Bytes preceding error
```

Obsérvese que LOAD no compone el valor check-sum en sí. Utilícese TYPE o un editor para inspeccionar el fichero HEX con el fin de ver exactamente lo que está equivocado.

CHECKSUM ERROR (PIP)

Si se pide al PIP que copie un fichero del tipo HEX, comprobará cada línea del fichero, asegurándose de que el control de suma de la línea es válido. Si no es así, PIP compondrá este mensaje. Desgraciadamente, PIP no comunica en qué línea está el error —esto deberá determinarse inspeccionando el fichero HEX o volviéndolo a crear e intentando de nuevo la orden.

COMMAND BUFFER OVERFLOW (SUBMIT)

SUBMIT compone este mensaje si el fichero SUB especificado es demasiado grande para ser tratado. La memoria intermedia interna de SUBMIT es sólo de 2.048 bytes. Hay que reducir el tamaño del fichero SUB; quitar las líneas de comentarios o dividirlas en dos ficheros con la última línea del primer fichero "sometiendo" al segundo para producir un fichero SUBMIT anidado.

COMMAND TOO LONG (SUBMIT)

La línea más larga de orden que puede procesar SUBMIT es de 125 caracteres. No hay otro medio de corregir este error que el de reducir la longitud de la línea que lo ha provocado. Habrá que encontrar esta línea mediante inspección —SUBMIT no la identifica.

Una forma para eliminar unos cuantos caracteres de una línea de orden es cambiar el nombre al fichero COM que se está invocando por un nombre más corto o utilizar nombres abreviados para parámetros si el programa los acepta.

CORRECT ERROR, TYPE RETURN OR CTL-Z (PIP)

Este mensaje es un remanente de los días en que PIP acostumbraba a leer datos hexadecimales de una lectora de cinta de papel de alta velocidad. Si PIP detectaba el final de un rollo físico de cinta de papel, componía este mensaje. El usuario podía entonces comprobar para ver si la cinta de papel se había roto o si efectivamente había llegado al final. Si había más cinta que leer, el usuario podía introducir un CARRIAGE RETURN para reanudar la lectura o introducir un CONTROL-Z para servir de carácter de final de fichero.

En pocas palabras, es poco probable que se presente este mensaje si no se tiene una lectora de cinta de papel.

DESTINATION IS R/O, DELETE (Y/N)? (PIP)

PIP compone este mensaje si se intenta escribir en un fichero de disco que haya sido fijado en modo de sólo lectura. Si se mecanografía "Y" o "y", PIP escribirá el fichero de destino. Deja el fichero de destino en modo de lectura/escritura con su estado directorio/sistema invariable. Al mecanografiar un carácter distinto de "Y" o "y" se hace que el PIP abandone la copia y componga el mensaje

```
** NOT DELETED **
```

Puede evitarse, sin embargo, este mensaje si se especifica la opción "w" en la línea de la orden PIP. Por ejemplo:

```
A>pip destfile=srcfile[w]<cr>
```

PIP escribirá entonces sobre los ficheros de sólo lectura sin preguntar.

DIRECTORY FULL (SUBMIT)

Este mensaje será compuesto si el BDOS devuelve un error cuando el SUBMIT intenta crear su fichero de salida, "$$$.SUB". Como aproximación rápida y general, úsese "STAT*.*" para ver cuántos ficheros y extensiones se tienen en el disco. Bórrense las que no se deseen. Usese entonces "STAT DSK:" para obtener el número máximo de posibles entradas al directorio para el disco.

También se puede ver este mensaje si el directorio de ficheros se ha deteriorado o si la rutina de formateado del disco abandona éste con el directorio de ficheros lleno con alguna configuración distinta de 0E5H.

Se puede apreciar si el directorio ha sido dañado utilizando "STAT USR:". STAT compone entonces los números de usuario que contiene el fichero. Si el directorio está dañado, normalmente se verán números de usuario mayores de 15.

No es fácil reparar un directorio estropeado. "ERA *.*" borra sólo los ficheros para el número de usuario en curso, por lo que habría que introducir la orden 16 veces, una para cada número de usuario de 0 a 15. También existe la alternativa de reformatear el disco.

DISK OR DIRECTORY FULL (ED)

Su mismo título indica claramente a qué se refiere.

DISK READ ERROR (PIP)
DISK WRITE ERROR (SUBMIT)
DISK WRITE ERROR (PIP)

Estos mensajes serán precedidos normalmente por un mensaje de error del BIOS. Sólo se visualizarán si el BIOS regresa indicando un error. Como se ha descrito anteriormente, esto no es probable si el BIOS tiene cualquier clase de lógica de recuperación de errores.

END OF FILE, CTL-Z? (PIP)

PIP compone este mensaje si, cuando copia un fichero HEX, encuentra un CONTROL-Z (final de fichero). De nuevo, la idea principal se basa en el concepto de la cinta de papel física. Cuando se veía este mensaje se podía mirar la cinta de papel en la lectora, y si realmente se había acabado el rollo, introducir un CONTROL-Z en el teclado para terminar el fichero. Al dar cualquier otro carácter, PIP habría leído el próximo trozo de cinta.

ERROR: CANNOT CLOSE FILES (LOAD)

LOAD compone este mensaje si se ha protegido físicamente contra la escritura el disco en el que se está intentando escribir la salida del fichero COM.

ERROR: CANNOT OPEN SOURCE (LOAD)

LOAD compone este mensaje si no puede abrir el fichero HEX que se ha especificado como parámetro de la orden.

ERROR: DISK READ (LOAD)
ERROR: DISK WRITE (LOAD)

Estos dos mensajes normalmente serán precedidos de un mensaje de error del BIOS. Si éste incluye recuperación de errores de disco, normalmente no se verá este mensaje; el error habrá sido tratado por el BIOS.

ERROR: INVERTED LOAD ADDRESS (LOAD)

LOAD compone este mensaje si detecta una dirección de carga menor que 0100H en el fichero HEX de entrada. También compone la dirección real de entrada desde el fichero, de forma que se pueda examinar el fichero HEX y buscar esta dirección para determinar la causa probable del problema.

Obsérvese que DDT, cuando se le pide que cargue el mismo fichero HEX, lo hará sin ningún error —y probablemente perjudique los contenidos de la página base al hacerlo.

ERROR: NO MORE DIRECTORY SPACE (LOAD)

Título expresivo por sí mismo.

ERROR ON LINE N (SUBMIT)

SUBMIT compone este mensaje si encuentra una línea en el fichero SUB que no sabe cómo procesar. Lo más probable es que se tenga un fichero con tipo .SUB pero que no contenga texto ASCII.

La primera línea del fichero SUB es la número 001.

FILE EXISTS (CCP)

El CCP compone este mensaje si se intenta utilizar la orden REN para cambiar el nombre a un fichero existente con un nombre ya dado a otro fichero.

Utilícese "STAT*.*" para visualizar todos los ficheros del disco. DIR

mostrará sólo aquellos que están en modo directorio y no se podrá ver el que causa el problema.

FILE IS READ/ONLY (ED)

ED compone este mensaje si se intenta editar un fichero que ha sido fijado a modo de sólo lectura.

FILE NOT FOUND (STAT)
FILENAME NOT FOUND (PIP)

STAT y PIP componen sus mensajes respectivos si se especifica un fichero no existente. Esto sirve tanto para nombres de fichero específicos como para ambiguos.

INVALID ASSIGNMENT (STAT)

STAT puede utilizarse para asignar dispositivos físicos a dispositivos lógicos utilizando el sistema IOBYTE descrito anteriormente. Compondrá este mensaje si se introduce una asignación ilógica. Utilícese la orden "STATVAL:" para visualizar la asignación válida.

INVALID CONTROL CHARACTER (SUBMIT)

Se supone que SUBMIT es capaz de manejar un carácter de control en el fichero SUB —la notación es "^x", siendo "x" la letra de control. De hecho, la versión estándar de SUBMIT no puede manejar esta notación. Existe una mejora disponible en Digital Research para corregir este problema.

Suponiendo que ésta haya sido instalada, SUBMIT compondrá este mensaje si se especifica un carácter distinto de "A" a "Z" después del acento circunflejo.

INVALID DIGIT (PIP)

PIP compone este mensaje si encuentra datos no numéricos donde espera un valor numérico.

INVALID DISK ASSIGNMENT (STAT)

STAT compone este mensaje si se intenta fijar un disco lógico en modo de sólo lectura y se especifica un parámetro distinto a "R/O". Obsérvese que no hay ningún "$" al principio en este caso (como lo hay cuando se desea fijar un fichero a modo de sólo lectura).

INVALID DRIVE NAME (USE A, B, C, OR D) (SYSGEN)

SYSGEN compone este mensaje si se intenta cargar el sistema CP/M desde una unidad de disco distinta de A, B, C o D, o bien escribir el sistema en dicho disco.

INVALID FILE INDICATOR (STAT)

STAT envía este mensaje si se especifica un atributo de fichero equivocado. Los atributos de fichero sólo pueden ser los siguientes:

$DIR	Directorio
$SYS	Sistema
$R/O	Sólo lectura
$R/W	Lectura/escritura

INVALID FORMAT (PIP)

PIP compone este mensaje si se introduce una orden formateada erróneamente; por ejemplo, un carácter "+" en lugar de un "=" (en algunos terminales están en la misma tecla).

INVALID HEX DIGIT (LOAD)

LOAD compone este mensaje si encuentra un dígito no hexadecimal en el fichero de entrada HEX, donde sólo puede aparecer un dígito hex. LOAD compone entonces información adicional para comunicar en qué lugar del fichero se produjo el problema:

```
INVALID HEX DIGIT
LOAD ADDRESS  0110  <- First address on line in file
ERROR ADDRESS 0112  <- Address of byte containing non-hex
BYTES READ:
0110:
0110: 00 33    <- Bytes preceding error
```

INVALID MEMORY SIZE (MOVCPM)

MOVCPM compone este mensaje si se introduce un tamaño de memoria no válido para el tamaño de sistema CP/M que se quiere construir.

INVALID SEPARATOR (PIP)

PIP compone este mensaje si se intenta concatenar ficheros usando algo distinto a una coma entre nombres de fichero.

INVALID USER NUMBER (PIP)

PIP compone este mensaje si se introduce un número de usuario fuera de los valores 0 a 15 con la opción "[gn]" (donde "n" es el número de usuario).

NO 'SUB' FILE PRESENT (SUBMIT)

SUBMIT compone este mensaje si no puede encontrar un fichero con el nombre especificado y tipo .SUB.

NO DIRECTORY SPACE (ASM)
NO DIRECTORY SPACE (PIP)

Títulos suficientemente claros.

NO FILE (CCP)

El CCP compone este mensaje si se usa la orden REN (cambio de nombre) y no puede encontrar el fichero que se desea nombrar.

NO FILE (PIP)

PIP compone este mensaje si no puede encontrar el fichero que se le ha especificado.

NO MEMORY (ED)

ED compone este mensaje si se sale de la memoria para almacenar el texto que se está editando.

NO SOURCE FILE ON DISK (SYSGEN)

Este mensaje de error puede inducir a equivocaciones. SYSGEN no lee ficheros en código fuente. El mensaje debería rezar "INPUT FILE NOT FOUND".

NO SOURCE FILE PRESENT (ASM)

En este caso, ASM indica realmente que el fichero en código fuente no puede encontrarse. Recuérdese que ASM utiliza una forma extraña para especificar sus parámetros. ASM usa el nombre de fichero que se introdujo y entonces busca un fichero de ese nombre, pero con tipo .ASM. Los tres caracteres del tipo de fichero que se especifiquen se utilizan para representar los discos lógicos en los cuales han de ser colocados el fichero fuente, el hex y los de listado, respectivamente.

NO SPACE (CCP)

El CCP compone este mensaje si se utiliza la orden SAVE y no hay sitio suficiente en el disco para acomodar el fichero.

NOT A CHARACTER SOURCE (PIP)

PIP compone este mensaje si se intenta copiar caracteres desde un dispositivo de salida de caracteres, tal como el dispositivo auxiliar de salida (conocido por PIP como PUN:).

OUTPUT FILE WRITE ERROR (ASM)

ASM compondrá este mensaje si el BDOS devuelve un error desde una operación de escritura de disco. Si el BIOS posee lógica de recuperación de errores de disco, no se verá nunca este mensaje.

PARAMETER ERROR (SUBMIT)

SUBMIT utiliza el "$" para señalar puntos donde los valores de parámetros han de ser sustituidos. Si se tiene un "$" simplemente seguido por un carácter alfabético, SUBMIT compondrá este mensaje. Utilícese "$$" para representar un "$" real.

PERMANENT ERROR, TYPE RETURN TO IGNORE (SYSGEN)

SYSGEN compone este mensaje si el BIOS devuelve un error desde la lectura de un disco o una operación de escritura. Si el BIOS posee lógica de recuperación de errores de disco, no debería verse nunca este mensaje.

QUIT NOT FOUND (PIP)

PIP compone este mensaje cuando no puede encontrar la cadena especificada en la opción "(Qcharacter string^Z)", que significa "Dejar de copiar cuando se encuentre esta cadena".

READ ERROR (CCP)

El CCP compone este mensaje si el BIOS devuelve un error desde la lectura de un disco o una operación de escritura. Si el BIOS incluye lógica de recuperación de errores de disco, no debería verse este mensaje de error.

RECORD TOO LONG (PIP)

PIP compone este mensaje si encuentra una línea mayor de 80 caracteres cuando copia un fichero HEX. Inspeccionar el fichero HEX utilizando la orden TYPE o un editor.

REQUIRES CP/M 2.0 OR NEWER FOR OPERATION (PIP)
REQUIRES CP/M VERSION 2.0 OR LATER (XSUB)

Mensajes suficientemente claros.

SOURCE FILE INCOMPLETE (SYSGEN)

SYSGEN compone este mensaje si el fichero que se ha solicitado que lea es demasiado corto. Utilícese STAT para comprobar la longitud del fichero.

SOURCE FILE NAME ERROR (ASM)

ASM compone este mensaje si se especifica un nombre de fichero ambiguo; es decir, uno que contenga "*" o "?".

SOURCE FILE READ ERROR (ASM)

ASM compone este mensaje si encuentra problemas al leer el fichero de entrada en código fuente. Compruébese el fichero de entrada utilizando la orden TYPE o un editor.

START NOT FOUND (PIP)

PIP compone este mensaje cuando no puede encontrar la cadena especificada en la opción "(Scharacter string^Z)", que significa "Empezar a copiar cuando encuentre esta cadena".

SYMBOL TABLE OVERFLOW (ASM)

ASM compone este mensaje cuando se tienen demasiados símbolos en el fichero de código fuente. El único recurso es dividir el fichero fuente en varias piezas y prepararlo para que las especificaciones ORG (de origen) coloquen el código objeto generado de forma que las piezas encajen juntas.

SYNCRONIZATION ERROR (MOVCPM)

Aparte del error de deletreado, este mensaje está diseñado para resultar críptico. MOVCPM compone este mensaje cuando el número de serie de Digital Research incluido en MOVCPM no coincide con el número de serie en la versión de CP/M que se está utilizando actualmente.

SYSTEM FILE NOT ACCESSIBLE (ED)

ED compone este mensaje si se intenta editar un fichero que ha sido fijado en modo sistema. Utilícese STAT para fijar el fichero en modo directorio.

TOO MANY FILES (STAT)

STAT compone este mensaje si hay insuficiente memoria disponible para ordenar y visualizar todos los ficheros del disco especificado. Inténtese

limitar el número de ficheros que tiene que ordenar mediante un uso juicioso de nombres de fichero ambiguos.

UNRECOGNIZED DESTINATION (PIP)

PIP compone este mensaje si se especifica un dispositivo de destino "ilegal".

VERIFY ERROR (PIP)

Si se utiliza la opción "[v]" (verificación) de PIP cuando se copia en un fichero de disco, PIP escribirá un sector del disco, lo leerá y comparará los datos. PIP compone este mensaje si los datos no corresponden.

Si existe un problema con el sistema de disco, se habrá visto alguna forma de mensaje de error del disco precediendo a éste. Si no hay mensaje precedente, entonces se tiene un problema con la memoria principal del sistema.

Wrong CP/M Version (Requires 2.0) (STAT)

Mensaje suficientemente claro.

(XSUB ACTIVE) (XSUB)

Este no es realmente un mensaje de error, pero se puede tomar por tal. XSUB es el programa SUBMIT extendido. Sin él, SUBMIT sólo puede alimentar líneas de órdenes al procesador de órdenes de consola. XSUB permite entradas carácter por carácter en cualquier programa que use el BDOS para leer entradas de consola.

XSUB se inicializa siendo la primera orden en un fichero SUB. Una vez iniciado permanece en memoria hasta que se haya alcanzado el final del fichero SUB. Hasta que esto ocurra, XSUB enviará este mensaje cada vez que ocurre un arranque caliente como recordatorio de que se encuentra aún en memoria.

XSUB Already Present (XSUB)

El XSUB compondrá este mensaje si está ya activo y se intenta cargarlo de nuevo.

Errores varios

Esta sección se enfrenta con los errores que no son acompañados por un mensaje de error. Se incluye aquí para ayudar a reconocer un problema después de que haya ocurrido.

Los errores se muestran agrupados por productos.

ASM: No se detecta el final de una operación IF

Si se utiliza la pseudooperación IF, debe seguirle un ENDIF. ASM omite detectar el caso en que el final del fichero fuente se encuentra *antes* del ENDIF.

Si la condición especificada en la línea IF es falsa, se puede tener una situación en la cual el ASM ignorará la mayoría del fichero fuente sin comentarios.

ASM: Crea un fichero HEX que no puede cargarse

Si se omite la especificación ORG al principio de un fichero fuente, el ASM ensamblará el código originado en el lugar 0000H. Este fichero hará caer el sistema si se intenta cargarlo con el DDT. El mensaje "ERROR: INVERTED ADDRESS" se mostrará desde el LOAD.

CP/M: Se conecta y luego desaparece sin indicación A>

Después de que el BIOS se ha conectado, transfiere el control al procesador de órdenes de consola. El CCP intenta entonces registrar en el disco del sistema, leyendo el directorio de ficheros y construyendo el vector de situación. Si el directorio de ficheros se ha dañado de forma importante, puede ocasionar el fallo de todo el sistema. Usese otro sistema de disco e inténtese visualizar el directorio del disco dañado.

DDT: Carga el fichero HEX y entonces cae el sistema

DDT no comprueba las direcciones especificadas en un fichero HEX. Si se ha olvidado poner una especificación ORG al principio del fichero fuente, o más sutilmente, si el programa fuente se ha "enrollado" por tener direcciones sobre 0FFFFH y "más", el ensamblador empezará ensamblando de nuevo en 0000H.

DIR: Muestra nombres de archivo raros

Si se tienen nombres de fichero que parecen raros o las líneas verticales de ":" que utiliza el DIR para separar los nombres de fichero están mal alineadas, entonces el directorio de ficheros se ha dañado. Una estrategia es formatear un nuevo disco, copiando todos los ficheros válidos en él, y descartar el disco estropeado.

DIR: Muestra más de una entrada con el mismo nombre

Esto puede ocurrir si se utiliza un programa que cree un fichero nuevo sin pedir al BDOS que borre cualquier otro existente del mismo nombre. También puede ocurrir si se utiliza descuidadamente la utilidad MOVE.

Para remediar la situación procédase en la forma siguiente:

- Usese PIP para copiar el fichero específico en otro disco. No debe usarse un nombre de fichero ambiguo; especifíquese el nombre del fichero duplicado exactamente. PIP copiará el primer fichero que encuentre en el directorio.
- Usese la orden ERA para borrar el fichero duplicado. *Esto borrará ambas copias del mismo.*
- Usese PIP para copiar el primero de los ficheros.

STAT: Números de usuario > 15

Si se usa la orden "STAT USR:" para visualizar qué números de usuario contienen ficheros activos y los números de usuario resultan mayores de 15, entonces el directorio de ficheros del disco está estropeado.

Utilícese PIP para copiar los ficheros válidos de los números de usuario legítimos y entonces deséchese el disco estropeado.

SUBMIT: Omite poner en marcha el procedimiento de sumisión

Hay varias razones por las que SUBMIT no inicia un fichero SUB:

- Se está utilizando la versión estándar de SUBMIT y el disco en curso por defecto es distinto del A:. SUBMIT construye su fichero "$$$.SUB" en el disco por defecto, pero el CCP sólo busca en la unidad A: "$$$.SUB". Utilícese el procedimiento siguiente para

modificar SUBMIT de forma que construya el fichero "$$$.SUB" en A:

```
A>DDT SUBMIT.COM<cr>
DDT VERS 2.2
NEXT  PC
0600 0100
-s5bb                  <- Change 5bb
05BB 01 00<cr>         <- from 00 (default drive)
05BC 24 .<cr>               to 01 (drive A:)
-^C
A>SAVE 5 SUBMIT.COM<cr>
A>_
```

- Si se olvida terminar la última línea del fichero SUB con un CARRIAGE RETURN.
- Si el fichero SUB contiene una línea con sólo un CARRIAGE RETURN (es decir, una línea en blanco).

Conjunto de caracteres ASCII

El código estándar americano para el intercambio de información (ASCII) consiste en un juego de 96 caracteres visibles y 32 no visibles. La mayoría de los sistemas CP/M utilizan por lo menos un subconjunto del conjunto de caracteres ASCII. Cuando el CP/M almacena caracteres en un disquete como texto, se utilizan las definiciones ASCII.

Algunos de los programas de utilidad del CP/M utilizan el código de caracteres ASCII. El texto creado utilizando el ED se almacena como caracteres ASCII en el disquete. El DDT, cuando visualiza un "volcado" de los contenidos de memoria, visualiza tanto la representación hexadecimal como la ASCII de los contenidos de la memoria.

El ASCII no utiliza un byte completo de información para representar un carácter. ASCII es un código de siete bits, y el octavo bit se utiliza a menudo para *paridad*. La paridad es un método de control de errores que asegura que el carácter recibido es el que se ha transmitido. Muchas microcomputadoras y sus dispositivos ignoran el *bit de paridad*, mientras que otras requieren una de las siguientes formas de paridad:

Paridad par

El número de 1s binarios de un byte es siempre un número par. Si hay un número impar de 1s en el carácter, el bit de paridad será

un 1; si hay un número par de 1s en el carácter, el bit de paridad se convierte en un 0.

Paridad impar

El número de 1s binarios de un byte es siempre un número impar. Si hay un número par de 1s en el carácter, el bit de paridad será un 1; si hay un número impar de 1s en el carácter, el bit de paridad se convierte en 0.

Otras formas alternativas de *codificar* la información almacenada por la computadora incluyen el EBCDIC de 8 bits (Código decimal de intercambio codificado en forma binaria extendida), utilizado por IBM y un número de esquemas *empaquetados en binario,* utilizados primariamente para representar información numérica.

Tabla A-1. Códigos de caracteres ASCII.

				b7 →	0	0	0	0	1	1	1	1
				b6 →	0	0	1	1	0	0	1	1
				b5 →	0	1	0	1	0	1	0	1
b4	b3	b2	b1	Fila \ Col.	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

NUL	Nulo	BEL	Timbre de alarma
SOH	Principio de encabezamiento	BS	Retroceso
STX	Principio de texto	HT	Tabulación horizontal
ETX	Final de texto	LF	Salto de línea
EOT	Final de transmisión	VT	Tabulación vertical
ENQ	Petición	FF	Salto de página
ACK	Contestación	CR	Retorno de carro

Tabla A-1. (Continuación.)

SO	Desplazamiento hacia fuera	CAN	Cancelación
SI	Desplazamiento hacia dentro	EM	Final del medio
DLE	Vínculo de escape de datos	SUB	Sustitución
DC1	Control de dispositivo 1	ESC	Escape
DC2	Control de dispositivo 2	FS	Separador de fichero
DC3	Control de dispositivo 3	GS	Separador de grupo
DC4	Control de dispositivo 4	RS	Separador de registro
NAK	Contestación negativa	US	Separador de unidad
SYN	Espera síncrona	SP	Espacio
ETB	Final del bloque de transmisión	DEL	Borrado

Tabla A-2. Códigos de carácter ASCII en orden ascendente.

Hexadecimal	Binario	ASCII	Hexadecimal	Binario	ASCII
00	000 0000	NUL	20	010 0000	SP
01	000 0001	SOH	21	010 0001	!
02	000 0010	STX	22	010 0010	"
03	000 0011	ETX	23	010 0011	#
04	000 0100	EOT	24	010 0100	$
05	000 0101	ENQ	25	010 0101	%
06	000 0110	ACK	26	010 0110	&
07	000 0111	BEL	27	010 0111	'
08	000 1000	BS	28	010 1000	(
09	000 1001	HT	29	010 1001	)
0A	000 1010	LF	2A	010 1010	*
0B	000 1011	VT	2B	010 1011	+
0C	000 1100	FF	2C	010 1100	,
0D	000 1101	CR	2D	010 1101	-
0E	000 1110	SO	2E	010 1110	.
0F	000 1111	SI	2F	010 1111	/
10	001 0000	DLE	30	011 0000	0
11	001 0001	DC1	31	011 0001	1
12	001 0010	DC2	32	011 0010	2
13	001 0011	DC3	33	011 0011	3
14	001 0100	DC4	34	011 0100	4
15	001 0101	NAK	35	011 0101	5
16	001 0110	SYN	36	011 0110	6
17	001 0111	ETB	37	011 0111	7
18	001 1000	CAN	38	011 1000	8
19	001 1001	EM	39	011 1001	9
1A	001 1010	SUB	3A	011 1010	:
1B	001 1011	ESC	3B	011 1011	;
1C	001 1100	FS	3C	011 1100	<
1D	001 1101	GS	3D	011 1101	=
1E	001 1110	RS	3E	011 1110	>
1F	001 1111	US	3F	011 1111	?

Tabla A-2. (Continuación.)

Hexadecimal	Binario	ASCII	Hexadecimal	Binario	ASCII
40	100 0000		60	110 0000	
41	100 0001	A	61	110 0001	a
42	100 0010	B	62	110 0010	b
43	100 0011	C	63	110 0011	c
44	100 0100	D	64	110 0100	d
45	100 0101	E	65	110 0101	e
46	100 0110	F	66	110 0110	f
47	100 0111	G	67	110 0111	g
48	100 1000	H	68	110 1000	h
49	100 1001	I	69	110 1001	i
4A	100 1010	J	6A	110 1010	j
4B	100 1011	K	6B	110 1011	k
4C	100 1100	L	6C	110 1100	l
4D	100 1101	M	6D	110 1101	m
4E	100 1110	N	6E	110 1110	n
4F	100 1111	O	6F	110 1111	o
50	101 0000	P	70	111 0000	p
51	101 0001	Q	71	111 0001	q
52	101 0010	R	72	111 0010	r
53	101 0011	S	73	111 0011	s
54	101 0100	T	74	111 0100	t
55	101 0101	U	75	111 0101	u
56	101 0110	V	76	111 0110	v
57	101 0111	W	77	111 0111	w
58	101 1000	X	78	111 1000	x
59	101 1001	Y	79	111 1001	y
5A	101 1010	Z	7A	111 1010	z
5B	101 1011	[	7B	111 1011	{
5C	101 1100	\	7C	111 1100	|
5D	101 1101	]	7D	111 1101	}
5E	101 1110	^	7E	111 1110	~
5F	101 1111	_	7F	111 1111	DEL

B

Resumen de órdenes CP/M

Este apéndice resume el formato de la línea de órdenes y la función de cada orden incorporada o transitoria de CP/M. Las órdenes figuran en la lista en orden alfabético.

Líneas de orden del ASM

ASM nombre-de-fichero<cr> Ensambla el fichero nombre-de-fichero.ASM; utiliza el disco registrado actualmente para todos los ficheros.

ASM nombre-de-fichero.opt<cr> Ensambla el fichero nombre-de-fichero.ASM en la unidad o: (A:,B:,...,P:). Escribe el fichero HEX en la unidad p: (A:,B:,...,P:), o salta si p: es Z:.

Escribe el fichero PRN en la unidad t: (A:, B:,...,P:), envía a la consola si p: es X:, o salta si p: es Z:.

Líneas de órdenes del DDT

DDT<cr> Carga el DDT y espera órdenes del mismo.

DDT x:nombre.typ<cr> Carga el DDT en la memoria y carga también nombre.typ desde la unidad x: en la memoria para examen, modificación o ejecución.

Resumen de órdenes del DDT

Assss Introduce sentencias en lenguaje ensamblador empezando en la dirección hexadecimal ssss.

D Visualiza los contenidos de los siguientes 192 bytes de memoria.

Dssss,ffff Visualiza los contenidos de memoria empezando en la dirección hexadecimal ssss y acabando en la dirección hexadecimal ffff.

Fssss,ffff,cc Llena la memoria con la constante hexadecimal de 8 bits cc empezando en la dirección hexadecimal ssss y acabando en la dirección hexadecimal ffff.

G Comienza la ejecución en la dirección contenida en el contador del programa.

G,bbbb Fija un punto de ruptura en la dirección hexadecimal bbbb, empezando entonces la ejecución en la dirección contenida en el contador del programa.

G,bbbb,cccc Fija un punto de ruptura en direcciones hexadecimales bbbb y cccc, empezando entonces la ejecución en la dirección contenida en el contador del programa.

Gssss Comienza la ejecución en la dirección hexadecimal ssss.

Gssss,bbbb Fija un punto de ruptura en la dirección hexadecimal bbbb y entonces empieza la ejecución en la dirección hexadecimal ssss.

Hx,y Suma y diferencia hexadecimales de x e y.

Inombre.typ Fija el bloque de control de fichero por defecto utilizando el nombre nombre-typ.

L Hace un listado de las siguientes once líneas del programa en lenguaje ensamblador desensamblado de la memoria.

Lssss Hace una lista de onde líneas del programa en lenguaje ensamblador desensamblado de la memoria empezando en la dirección hexadecimal ssss.

Lssss,ffff Hace una lista del programa en lenguaje ensamblador desensamblado de la memoria empezando en la dirección hexadecimal ssss y acabando en la dirección hexadecimal ffff.

Mssss,ffff,dddd Traslada los contenidos del bloque de memoria empezando en la dirección hexadecimal ssss y acabando en la dirección hexadecimal ffff al bloque de memoria que empieza en la dirección hexadecimal dddd.

R Lee un fichero del disco en la memoria (utilícese antes la orden "I").

Rnnnn Lee un fichero del disco en la memoria empezando en la dirección hexadecimal nnnn mayor que la normal (úsese antes la orden "I").

Sssss Visualiza los contenidos de la memoria en la dirección hexadecimal ssss y cambia opcionalmente los contenidos.

Tnnnn Realiza la ejecución de nnnn instrucciones del programa (hexadecimal).

Unnnn Ejecuta nnnn instrucciones del programa (hexadecimal), deteniéndose entonces y visualizando los contenidos de los registros de la CPU.

X Visualiza los contenidos de los registros de la CPU.

Xr Visualiza los contenidos de la CPU o del indicador r y opcionalmente los cambia.

Líneas de orden DIR

DIR x:<cr> Visualiza el directorio de todos los ficheros en la unidad x:. La unidad x: es opcional; si se omite, será utilizada la unidad en curso.

DIR x:nombre.typ<cr> Visualiza el directorio de todos los ficheros en la unidad x: cuyos nombres coinciden con el nombre de fichero.tipo sea o no ambiguo. La especificación de unidad x: es opcional; si se omite, se utiliza la unidad en curso.

Línea de orden DUMP

DUMP x:nombre.typ<cr> Visualiza las representaciones hexadecimales de cada byte almacenado en el fichero nombre.typ en la unidad x:. Si el

nombre.typ es ambiguo, visualiza el primer fichero que concuerde con el nombre ambiguo.

Línea de orden ED

ED x:nombre.typ<cr> Invoca al editor, quien busca entonces el fichero nombre.typ en la unidad x: y crea un fichero temporal x:nombre.$$$ para almacenar el texto editado. El nombre.typ no es ambiguo. La unidad x: es opcional; si se omite, se asume la unidad actualmente en curso.

Resumen de órdenes del ED

NOTA: Las órdenes no alfabéticas siguen a la orden "Z".

nA Añade líneas. Traslada "n" líneas del fichero original a la memoria intermedia de edición. 0A traslada líneas hasta que la memoria de edición está por lo menos llena a mitad.

+/B Comienzo/Final. Traslada el CP.
+B traslada el CP al principio de la memoria de edición
−B traslada el CP al final de la memoria de edición

+/−nC Traslada por caracteres. Traslada el CP "n" posiciones de caracteres.
+ traslada hacia delante
− traslada hacia atrás

+/−nD Borra caracteres. Borra "n" caracteres antes o después del CP en la memoria de edición.
+ borra antes del CP
− borra después del CP

E Termina. Termina la edición, cierra ficheros y vuelve al CP/M; final normal.

nFcadena^Z Encontrar cadena. Encuentra la "enésima" ocurrencia de la cadena, empezando la búsqueda después del CP.

H Traslada a la cabeza del fichero editado. Acaba la edición, vuelve a nombrar los ficheros y edita entonces los primeros ficheros temporales.

I<cr> Introduce el modo de inserción. El texto del teclado va a la memoria intermedia de edición después del CP; salida con CONTROL-Z.

Icadena^Z Inserta cadena. Inserta una cadena en la memoria de edición detrás del CP.

Icadena<cr> Inserta línea. Inserta cadena y CRLF en la memoria de edición después del CP.

nJhallarcadena^Zinsertarcadena^Zfinalcadena^Z Yuxtaposición. Comienza después del CP, encuentra "hallar", inserta "insertar" después de ésta y borra entonces todos los caracteres siguientes, pero no incluyendo "final" repite hasta haberlo realizado "n" veces.

+/−nK Mata líneas. Borra "n" líneas.
+ borra después de CP
− borra antes del CP

+/−nL Traslada por líneas. Traslada el CP al principio de la línea en la que está, entonces traslada el CP "n" líneas hacia delante o hacia atrás.
+ traslada hacia delante
− traslada hacia atrás

nMcadenaorden^Z Macroorden. Repite la ejecución de las órdenes ED "cadena-orden" "n" veces. "n"=0, "n"=1, o en ausencia de "n" repite la ejecución hasta que ocurra un error.

nNcadena^Z Encuentra la cadena con autorregistro. Encuentra la "enésima" ocurrencia de la misma, añadiendo automáticamente del fichero original y escribiendo en el fichero temporal si es necesario.

O Vuelve al fichero original. Vacía la memoria de edición, vacía el fichero temporal, vuelve al principio del fichero original, ignora las órdenes ED previas.

+/−nP Traslada el CP e imprime páginas. Traslada el CP hacia delante o hacia atrás una página y visualiza entonces la página siguiente al CP. "nP" visualiza "n" páginas, haciendo una pausa después de cada una.

Q Detiene la edición. Borra el fichero y el bloque temporales, traslada archivo, si existe, y vuelve al CP/M; el fichero original no se cambia.

R<cr> Lee el archivo de traslado de bloque. Copia todo el fichero de traslado de bloque X$$$$$$$.LIB desde el disco y lo inserta en la memoria intermedia de edición después del CP.

Rnombre<cr> Lee el fichero de biblioteca. Copia todo el fichero "nombre" con extensión LIB desde el disco y lo inserta en la memoria de edición después del CP.

nScadena1^Zcadena2^Z Sustituye cadena. Empezando en el CP, repite "n" veces: encuentra cadena1 y la reemplaza por cadena2.

+/−nT Mecanografía líneas. Visualiza "n" líneas.
+ "n" líneas después del CP
− "n" líneas antes del CP

Si el CP no está al principio de una línea:
0T visualiza desde el principio de la línea hasta el CP
T visualiza desde el CP hasta el final de la línea
0TT visualiza toda la línea sin desplazar el CP

+/−U Traducción a mayúscula. Después de una orden +U, la entrada alfabética a la memoria de edición se traduce de minúsculas a mayúsculas; después de −U, no se realiza ninguna traducción.

OV Espacio/tamaño libre de la memoria de edición. Visualiza el número decimal de bytes libres (vacíos) en la memoria de edición y el tamaño total de la misma.

+/−V Verifica números de línea. Después de +V, se visualiza un número de línea con cada línea. La indicación ED va entonces precedida por el número de la línea que contiene el CP. Después de −V, los números de línea no se visualizan y la indicación ED es "*".

nW Escribe líneas. Escribe primero "n" líneas desde la memoria de edición en el fichero temporal; borra estas líneas de la memoria de edición.

nX Transfiere bloque (Xfer). Copia las "n" líneas que siguen al CP desde la memoria de edición en el fichero de traslado de bloques temporal X$$$$$$$.LIB; lo añade a los contenidos previos de dicho fichero.

nZ Duerme. Se retrasa en la ejecución de la orden que le sigue. Cuanto mayor sea "n" mayor será el retraso; los "n" reducidos dan retrasos menores.

n: Traslada el CP al número de línea "n". Traslada el CP al principio del número de línea "n" (*véase* "+/−V").

:m Continúa hasta el número de línea "m". Un prefijo de orden que da el punto final para la orden que le sigue. El punto de comienzo es la situación del CP (*véase* "+/−V").

+/−n Traslada y visualiza una línea. Abreviatura de +/−nLT.

Líneas de orden ERA

ERA x:nombre.typ<cr> Borra el fichero nombre.typ del disco en unidad x:. El nombre y/o el tipo pueden ser ambiguos. La unidad x: es opcional; si se omite, se utiliza la unidad en curso actual.

ERA x:*.*<cr> Borra todos los ficheros del disco en unidad x:. La unidad x: es opcional; si se omite, se utiliza la unidad en curso actual.

Ordenes de edición de línea

CONTROL-C Vuelve a poner en marcha el CP/M si éste es el primer carácter de la línea de mando. Denominado *arranque en caliente.*

CONTROL-E Traslada al comienzo de la línea siguiente. Utilizado para mecanografiar órdenes largas.

CONTROL-H o BACKSPACE Borra un carácter y lo elimina de la pantalla (versiones 2.0 y posteriores del CP/M).

CONTROL-J o LINE FEED Lo mismo que CARRIAGE RETURN (versiones 2.0 y posteriores del CP/M).

CONTROL-M Lo mismo que CARRIAGE RETURN (<cr>).

CONTROL-P Conecta el dispositivo de listado (usualmente la impresora). Púlsese otra vez para desconectar el dispositivo de listado.

CONTROL-R Repite la línea de orden actual (útil con la versión 1.4); verifica que la línea esté corregida después de haber borrado varios caracteres (versiones 1.4 y posteriores del CP/M).

CONTROL-S Detiene temporalmente la visualización de datos en la consola. Púlsese cualquier tecla para continuar.

CONTROL-Y o CONTROL-X Borra la línea de orden actual (versiones 1.4 y posteriores del CP/M).

RUBOUT (RUB) o DELETE (DEL) Borra un carácter y lo repite.

Línea de orden de carga

LOAD x:nombre<cr> Lee el fichero.HEX en la unidad x: y crea el fichero de programa ejecutable nombre.COM en la unidad x:.

Líneas de orden MOVCPM

MOVCPM<cr> Prepara una nueva copia del CP/M que utiliza toda la memoria; da el control al nuevo CP/M, pero no lo salva en el disco.

MOVCPM nn<cr> Prepara una nueva copia del CP/M que utiliza "nn" Kbytes de memoria; da el control al nuevo CP/M, pero no lo salva en el disco.

MOVCPM * * <cr> Prepara una nueva copia del CP/M que utiliza toda la memoria para ser salvada con SYSGEN o SAVE.

MOVCPM nn * <cr> Prepara una nueva copia del CP/M que utiliza "nn" Kbytes de memoria, a salvar con SYSGEN o SAVE.

"nn" es un número decimal. Puede ser de 16 a 64 para el CP/M 1.3 o 1.4. Para el CP/M 2.0 y posteriores "nn" puede ser 20 a 64.

Líneas de orden PIP

PIP<cr> Carga el PIP en memoria. PIP indica en la consola las órdenes, las ejecuta y entonces vuelve a indicarlo en la consola.

PIP línea-de-orden-PIP<cr> Carga el PIP en la memoria. El PIP ejecuta la orden línea-de-orden-PIP y después sale al CP/M.

Resumen de órdenes del PIP

x:nuevo.typ=y:viejo.typ[p]<cr> Copia el fichero viejo.typ en la unidad y: en el fichero nuevo.typ en la unidad x:, utilizando parámetros p.

x:nuevo.typ=y:viejo1.typ[p],z:viejo2.typ[q]<cr> Crea un fichero nuevo.typ en la unidad x: que consta del contenido del fichero viejo1.typ en la unidad y: utilizando parámetros p seguidos por los contenidos del fichero viejo2.typ de la unidad z: usando parámetros q.

x:nombre.typ=dev:[p]<cr> Copia datos del dispositivo dev: en el fichero nombre.typ de la unidad x:.

dev:=x:nombre.typ[p]<cr> Copia datos del nombre.typ en unidad x: en el dispositivo dev:.

dst:=src:[p]<cr> Copia datos en el dispositivo dst: del dispositivo src:.

Resumen de los parámetros de PIP

B	Especifica transferencia modo bloque.
Dn	Borra todos los caracteres después de la "enésima" columna.
E	Refleja la copia en la consola según se va realizando.
F	Elimina los caracteres de salto de línea durante la transferencia.
Gn	Dirige a PIP a copiar un fichero desde el área de usuario "n".
H	Controla el formato de fichero Hex Intel adecuado.
I	Ignora todos los registros *:00* de las transferencias de fichero Hex Intel.
L	Traduce las letras mayúsculas a minúsculas.
N	Añade un número de línea a cada línea transferida.
O	Se opone a la transferencia de fichero (ignora los indicadores de final de fichero).
Pn	Lanza un salto de página después de cada "n" líneas.
Qs^Z	Especifica el cese de copiado después de encontrar la cadena "s".
R	Dirige a PIP a copiar de un fichero de sistema.
Ss^Z	Especifica el principio de copiado después de haber encontrado la cadena "s".
Tn	Fija los topes de tabulación en cada "n" columnas.
U	Traduce las letras minúsculas a mayúsculas.
V	Verifica las copias por comparación después de haber terminado de copiar.
W	Dirige a PIP para copiar en un fichero R/O.
Z	Pone a cero el bit de "paridad" en los caracteres ASCII.

Dispositivos de destino de PIP

CON:	PUN:	LST:	Dispositivos lógicos
TTY:	PTP:	LPT:	
CRT:	UP1:	UL1:	
UC1:	UP2:		Dispositivos físicos
OUT:	PRN:		Dispositivos especiales PIP

Dispositivos fuente de PIP

CON:	RDR:		Dispositivos lógicos
TTY:	PTR:		
CRT:	UR1:		
UC1:	UR2:		Dispositivos físicos
NUL:	EOF:	INP:	Dispositivos especiales PIP

Línea de orden REN

REN nuevo-nombre.typ=nombre-viejo.typ<cr> Encuentra el fichero nombre-viejo-typ y le cambia el nombre a nuevo-nombre-typ.

Línea de orden SAVE

SAVE nnn x:nombre.typ<cr> Salva una porción del área de programa transitorio de la memoria en el fichero nombre.typ de la unidad x:, donde nnn es un número decimal que representa el número de páginas de la memoria. x: es el especificador de unidad opcional.

Líneas de orden STAT

STAT<cr> Visualiza atributos y la cantidad de espacio libre para todas las unidades de disquete a las que se accede desde el último arranque caliente o frío.

STAT x:<cr> Visualiza la cantidad de espacio libre del disquete en la unidad x:.

STAT x:nombre.typ<cr>(CP/M 2.0 y posteriores) Visualiza tamaño y atributos del fichero(s) nombre.typ en la unidad x:. nombre.typ puede ser ambiguo. x: es opcional; si se omite, se supone unidad en curso.

STAT x:nombre.typ $atr<cr> Asigna el atributo atr al fichero(s) nombredefichero.tipo en la unidad x:. Nombredefichero.tipo puede ser ambiguo. La especificación de unidad x: es opcional; si se omite, se utiliza la unidad en curso.

STAT DEV:<cr> Informa de los dispositivos físicos que están asignados actualmente a los cuatro dispositivos lógicos.

STAT VAL:<cr> Informa sobre las posible asignaciones de dispositivos y sobre el resumen de la línea de orden parcial STAT.

STAT log:=phy:<cr> Asigna el dispositivo físico phy: al dispositivo lógico log. (puede haber más de una asignación en la línea; cada una deberá estar separada por una coma).

STAT USR:<cr> (CP/M 2.0 y posteriores) Informa sobre el número de usuario en curso, así como de los números de usuario para los cuales hay ficheros en los discos actualmente en curso.

STAT x:DSK<cr> (CP/M 1.4 y posteriores) Asigna un modo temporal de protección de escritura a la unidad x:.

Líneas de orden SUBMIT

SUBMIT nombre<cr> Crea un fichero $$$.SUB que contiene las órdenes listadas en nombre.SUB; el CP/M ejecuta entonces órdenes de este fichero antes que del teclado.

SUBMIT nombre parámetros<cr> Crea un fichero $$$.SUB que contiene órdenes del fichero nombre.SUB; ciertas partes de las líneas de orden del fichero nombre.SUB son reemplazadas por parámetros durante la creación de $$$.SUB. CP/M obtiene entonces órdenes de este fichero mejor que del teclado.

Línea de orden SYSGEN

SYSGEN<cr> Carga el programa SYSGEN para transferir el CP/M de un disquete a otro.

Línea de orden TYPE

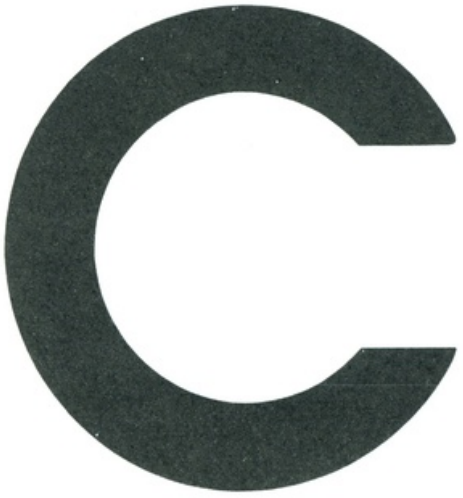
TYPE x:nombre.typ<cr> Visualiza los contenidos del fichero nombre.typ de la unidad x: en la consola.

Línea de orden USER

USER n<cr> Fija el número de usuario en "n", donde "n" es un número decimal de 0 a 15, inclusive.

Línea de orden x:

x:<cr> Cambia la unidad de disco actualmente en curso a la unidad x:. La unidad x: puede ser de "A" a "P".



Resumen de las llamadas del BDOS

Tabla C-1. Definiciones de función del BDOS para la versión 2.2 del CP/M-80.

Función		Parámetro(s) de entrada	Parámetro(s) de salida	Aclaración
N.º	Nombre			
00	SYSTEM RESET	Ninguno	Ninguno	Nueva puesta en marcha del CP/M-80 devolviendo el control al CCP después de reinicializar el subsistema del disco.
01	CONSOLE INPUT	Ninguno	A=Carácter ASCII	Devuelve el siguiente carácter mecanografiado al programa de llamada de caracteres. Cualquier carácter no imprimible se refleja en la pantalla (como BACKSPACE, TAB o CARRIAGE RETURN). La ejecución no vuelve al programa de llamada hasta que se ha mecanografiado un carácter. Los caracteres de control del CCP estándar son reconocidos y sus acciones realizadas (CONTROL-P empieza o termina la impresión, etc.).

Tabla C-1. (Continuación.)

Función		Parámetro(s) de entrada	Parámetro(s) de salida	Aclaración
N.°	Nombre			
02	CONSOLE OUTPUT	E=Carácter ASCII	Ninguno	Visualiza el carácter en el registro E en el dispositivo de consola. Los caracteres de control del CCP estándar son reconocidos y sus acciones realizadas (CONTROL-P empieza o termina la impresión, etc.).
03	READER INPUT	Ninguno	A=Carácter ASCII	Devuelve el próximo carácter que se recibe del dispositivo de lectura al programa de llamada. La ejecución no vuelve al programa de llamada hasta que se recibe un carácter.
04	PUNCH OUTPUT	E=Carácter ASCII	Ninguno	Transmite el carácter del registro E al dispositivo de perforación.
05	LIST OUTPUT	E=Carácter ASCII	Ninguno	Transmite el carácter del registro E al dispositivo de listado.
06	DIRECT CONSOLE IN DIRECT CONSOLE OUT	E=FF hex E=Carácter ASCII	A=ASCII Ninguno	Si el registro E contiene un hex FF, el dispositivo de consola es interrogado para ver si un carácter está preparado. Si no hay ninguno, se devuelve un 00 al programa de llamada en el registro A, en caso contrario el carácter detectado se devuelve al registro A. Si el registro E contiene cualquier carácter distinto de FF hex, aquel carácter es transferido para visualización en consola. Todos los caracteres de control CCP se ignoran. El usuario debe proteger el programa contra caracteres disparatados que sean enviados o recibidos por el dispositivo de consola.
07	GET IOBYTE	Ninguno	A=IOBYTE	Coloca una copia del byte almacenado en la situación 0003 hex del registro A antes de devolver el control al programa de llamada.
08	SET IOBYTE	E=IOBYTE	Ninguno	Coloca una copia del valor del registro E en la posición de memoria 0003 hex antes de devolver el control al programa de llamada.
09	PRINT STRING	DE=Dirección de hilera	Ninguno	Envía la cadena de caracteres almacenados empezando en la dirección almacenada en el par de registro DE al dispositivo de consola. Todos los caracteres en las direcciones siguientes se envían hasta que el BDOS encuentra una posición de memoria que contenga un 24 hex (un "$" ASCII). Los caracteres de control del CCP son controlados y realizados si se encuentran.

NOTA: El CP/M-80 copia siempre los contenidos del registro H en el registro A si no hay que devolver nada específicamente al registro A. Algunos fabricantes, especialmente Microsoft, utilizan tal información para reducir el movimiento de información entre los registros H y A.

Tabla C-1. (Continuación.)

Función		Parámetro(s) de entrada	Parámetro(s) de salida	Aclaración
N.º	Nombre			
0A	READ CONSOLE BUFFER	DE=Dirección de la memoria intermedia	Datos en la memoria intermedia	Esta función realiza esencialmente la misma que el CCP haría, en que toma los caracteres que mecanografía el usuario y los almacena en la memoria que empieza en la dirección almacenada en el par de registro DE. El primer byte en la memoria apuntado por el par DE debe ser la longitud máxima de la orden; el BDOS colocará el número de caracteres encontrados en el segundo byte, con la orden mecanografiada empezando con el tercer byte apuntado por el par DE. Todos los caracteres de edición del CCP estándar son reconocidos durante la entrada de la orden.
0B	GET CONSOLE STATUS	Ninguno	A=Estado	El BDOS controla el estado del dispositivo de consola y devuelve un 00 hex si no hay ningún carácter preparado, y FF hex si se ha mecanografiado un carácter.
0C	GET VERSION NUMBER	Ninguno	HL=Versión	Si el byte devuelto en el registro H es 00 hex, entonces el CP/M está presente; si es 01, entonces está presente el MP/M. El byte devuelto en el registro L es 00 si la versión es anterior al CP/M 2.0, 20 hex si la versión es 2.0, 21 hex si es 2.1, etcétera.
0D	RESET DISK SYSTEM	Ninguno		Utilizado para indicar al CP/M que ponga a cero el subsistema de disco. Debería utilizarse cada vez que se cambian los disquetes.
0E	SELECT DISK	E=Número de disco	Ninguno	Selecciona el disco que ha de ser utilizado para operaciones de disco subsiguientes. Un 00 hex en el registro E indica disco A, un 01 hex indica disco B, etc.
0F	OPEN FILE	DE=Dirección FCB	A=Código de "encontrado" no encontrado	Utilizado para activar un fichero de la unidad de disco actual y del área de usuario actual. El BDOS registra los primeros 14 bytes del bloque FCB designado e intenta encontrar un equivalente al nombre en el bloque. Un 3F hex ("?" ASCII) puede utilizarse en cada una de las posiciones del nombre para indicar un carácter cualquiera. Si se encuentra un equivalente, la información relevante acerca de este fichero se introduce en el resto del FCB por el CP/M-80. Un valor 00 hex a 03 en el registro A al retorno indica que la operación de apertura ha sido satisfactoria, mientras que un FF hex indica que no se ha podido encontrar el fichero. Si se utilizan signos interrogativos para identificar un fichero, se utiliza la primera entrada equivalente.

NOTA: El CP/M-80 copia siempre los contenidos del registro H en el registro A si no hay que devolver nada específicamente al registro A. Algunos fabricantes, especialmente Microsoft, utilizan tal información para reducir el movimiento de información entre los registros H y A.

Tabla C-1. (Continuación.)

Función		Parámetro(s) de entrada	Parámetro(s) de salida	Aclaración
N.º	Nombre			
10	CLOSE FILE	DE=Dirección FCB	A=Código de "encontrado"/no encontrado	Realiza lo contrario de la función de apertura de fichero. Una operación de cerrar fichero debe ser realizada después de cada utilización de un fichero que contiene en su interior información escrita.
11	SEARCH FOR FIRST	DE=Dirección FCB	A=Código de "encontrado"/no encontrado	Realiza lo mismo que la función de apertura de fichero, con la diferencia de que la memoria intermedia de disco se llena con el registro de 128 bytes que es la entrada de directorio del fichero comparado.
12	SEARCH FOR NEXT	Ninguno	A=Código de "encontrado"/no encontrado	Realiza lo mismo que la función búsqueda del primer caso, excepto que la búsqueda continúa desde la última entrada comparada.
13	DELETE FILE	DE=Dirección FCB	A=Código de "encontrado"/no encontrado	Cambia un indicador en la entrada para el fichero apuntado por el FCB de forma que el CP/M-80 no siga reconociéndolo como un fichero válido. No se borra prácticamente ninguna información cuando se realiza esta función, aunque escrituras subsiguientes en el disquete utilizarán el área previamente asociada al fichero "borrado".
14	READ SEQUENTIAL	DE=Dirección FCB	A=Código de error	Si un fichero ha sido activado para su uso por la función de apertura o de creación de fichero, la función de lectura secuencial lee el próximo bloque de 128 bytes en la memoria en la dirección actual de DMA. El valor 00 hex se devuelve al registro A si la lectura ha sido satisfactoria, mientras que un valor distinto a cero en el registro A indica fracaso.
15	WRITE SEQUENTIAL	DE=Dirección FCB	A=Código de error	Si se ha activado un fichero para realizar una función de apertura o una función fichero, la función de escritura secuencial escribe el bloque de 128 bytes que se encuentra en la dirección de DMA en curso, en el registro de 128 bytes que sigue al fichero especificado.
16	MAKE FILE	DE=Dirección FCB	A=Código DIR	Crea un nuevo fichero con la información (nombre) indicada por el FCB. El CP/M-80 no controla si el fichero indicado existe ya, de modo que primero hay que comprobar si existe el fichero (o borrarlo). Un fichero creado de nuevo no necesita ser abierto, porque la función de crear fichero también realiza las operaciones de apertura necesarias.

NOTA: El CP/M-80 copia siempre los contenidos del registro H en el registro A si no hay que devolver nada específicamente al registro A. Algunos fabricantes, especialmente Microsoft, utilizan tal información para reducir el movimiento de información entre los registros H y A.

Tabla C-1. (Continuación.)

Función		Parámetro(s) de entrada	Parámetro(s) de salida	Aclaración
N.º	Nombre			
17	RENAME FILE	DE=Dirección FCB	A=Código DIR	Cambia el nombre del fichero referenciado por los primeros 16 bytes del FCB al nombre de los segundos 16 bytes.
18	RETURN LOGIN VECTOR	Ninguno	HL=Disco en curso	Los bits del registro HL se utilizan para especificar las unidades de disco que están activas. El primer bit del registro L se refiere a la unidad A, el último bit del registro H corresponde a la unidad P, la más alta posible. Un valor 1 del bit indica unidad activa, un cero denota unidad inactiva.
19	RETURN CURRENT DISK	Ninguno	A=Disco actual	Los números 0 a 15 son utilizados para representar la unidad de disco por defecto en curso como retorno de esta función.
1A	SET DMA ADDRESS	DE=DMA	Ninguno	Utilizado para seleccionar el bloque de memoria de 128 bytes que ha de utilizarse como memoria intermedia en todas las transferencias de disco. Sobre la puesta a cerco del sistema o del disco, fría o caliente, la memoria intermedia se sitúa en 0080 hex en un sistema CP/M normal.
1B	GET ALLOC ADDRESS	Ninguno	HL=Dirección de ubicación	Devuelve la dirección de comienzo del vector de ubicación, una tabla que indica las posiciones del disquete que se utilizan y que es mantenida en memoria para cada unidad de disco en línea.
1C	WRITE PROTECT DISK	Ninguno	Ninguno	Procura protección de escritura temporal para el disquete en la unidad de disco por defecto en curso.
1D	GET R/O VECTOR	Ninguno	HL=R/O de disco	Devuelve un valor de 16 bits en los registros HL, lo que indica las unidades del sistema cuya escritura está protegida. Las unidades están asignadas como en el VECTOR LOGIN, con un valor 1 indicando protección de escritura
1E	SET FILE ATTRI-BUTES	DE=Dirección FCB	A=Código DIR	Fija los atributos de fichero que indican sistema/directorio y estado R/O o R/W para el fichero apuntado por la dirección del FCB.
1F	GET DISK PARMS	Ninguno	HL=Dirección DPB	Recupera el bloque de parámetros del disco para la unidad de disco activa en curso. Estos parámetros pueden utilizarse para determinar el espacio disponible en un disquete o para cambiar las características de la unidad de disco bajo control del usuario.

NOTA: El CP/M-80 copia siempre los contenidos del registro H en el registro A si no hay que devolver nada específicamente al registro A. Algunos fabricantes, especialmente Microsoft, utilizan tal información para reducir el movimiento de información entre los registros H y A.

Tabla C-1. (Continuación.)

Función		Parámetro(s) de entrada	Parámetro(s) de salida	Aclaración
N.º	Nombre			
20	GET USER CODE SET USER CODE	E=FF E=Código de usuario	A=Usuario actual o ninguno	Si el registro E contiene un FF hex, el número de usuario se devuelve en el registro A. Para volver a fijar el número de usuario, el código de usuario adecuado está situado en el registro E. Mientras que la orden USER permite números de usuario de 0-15, esta función del BDOS puede fijar números de usuario de 0-31.
21	READ RANDOM	DE=Dirección FCB	A=Código de error	Lee el número de registro aleatorio contenido en los bytes 33.º, 34.º y 35.º (una direccion de 24 bits) del FCB indicado.
22	WRITE RANDOM	DE=Dirección FCB	A=Código de error	Escribe información de la dirección de DMA actual en el registro aleatorio apuntado por el número contenido en los bytes 33.º, 34.º y 35 del FCB indicado.
23	COMPUTE FILE SIZE	DE=Dirección FCB	Conjunto RRF	Devuelve el tamaño actual del fichero de registro aleatorio en los tres bytes que constituyen el campo de registro aleatorio del FCB. Si el tercer byte contiene un 1, entonces el fichero contiene el valor máximo de registro de 65.536, en otro caso el valor de los dos primeros bytes es un valor de 16 bits que representa el tamaño del fichero.
24	SET RANDOM RECORD	DE=Dirección FCB	Conjunto RRF	Devuelve registro aleatorio siguiente (llena el campo de registro aleatorio del FCB) después del último registro leído secuencialmente. Digital Research sugiere que esta función es más adecuada para indexado de ficheros.
25	RESET DRIVE	DE=Bits de puesta a cero de impulsor	A=Código de error	Fuerza a las unidades especificadas a poner los bits de control a su estado inicial no en línea.
28	WRITE RANDOM (ZERO)	DE=Dirección FCB	A=Código de error	Escribe un registro de todo ceros en el disquete antes de que se escriba un registro; útil para identificar registros aleatorios no utilizados (un registro no utilizado contendrá ceros en lugar de datos).

NOTA: El CP/M-80 copia siempre los contenidos del registro H en el registro A si no hay que devolver nada específicamente al registro A. Algunos fabricantes, especialmente Microsoft, utilizan tal información para reducir el movimiento de información entre los registros H y A.

D

Resumen de las llamadas del BIOS

Tabla D-1. Definiciones de rutina BIOS del CP/M-80.

Etiqueta de la tabla de salto	Parámetro(s) de entrada	Parámetro(s) de salida	Aclaración
COLDSTART	Ninguno	C=0	La rutina deberá realizar todas las operaciones necesarias de puesta en marcha, incluida la inicialización de todos los valores de la página base. Antes de salir, el registro C deberá ser puesto a cero.
WARMSTART	Ninguno	C=Unidad	La rutina deberá realizar todas las operaciones necesarias de nueva puesta en marcha, pero no necesita reinicializar la página base. El registro C, a la salida, debería contener el número de unidad en curso.
CONSOLE STATUS (CONST)	Ninguno	A=Estado	

Tabla D-1. (Continuación.)

Etiqueta de la tabla de salto	Parámetro(s) de entrada	Parámetro(s) de salida	Aclaración
CONSOLE* INPUT	Ninguno	A=Carácter	
READER* INPUT	Ninguno	A=Carácter	La rutina debería esperar un carácter que se ha de introducir en el dispositivo adecuado y entonces devolver el carácter en el registro A.
CONSOLE* OUTPUT	C=Carácter	Ninguno	
LIST* OUTPUT	C=Carácter	Ninguno	
PUNCH* OUTPUT	C=Carácter	Ninguno	La rutina debería tomar el carácter en el registro C y visualizarlo en el dispositivo adecuado.
HOME DISK	Ninguno	Ninguno	La cabeza de la unidad de disco debería ser devuelta a la posición inicial (pista 0, sector 0).
SELECT DISK	C=Unidad	HL=DHA	La rutina debería seleccionar la unidad indicada por el número en el registro C. El registro HL al regresar debería contener la dirección de la cabecera de parámetros del disco.
SET TRACK	C=Pista	Ninguno	La pista indicada por el valor del registro C debería ser fijada en la pista siguiente para que acceda a ella el controlador de disco.
SET SECTOR	C=Sector	Ninguno	El sector indicado por el valor del registro C debería ser fijado en la pista siguiente para que acceda a él el controlador de disco.
SET DMA ADDRESS	BC=Dirección DMA	Ninguno	La dirección de DMA indicada por el par de registros BC debería ser fijada como la dirección para usar todas las transferencias de información desde la memoria al disquete y viceversa.
READ DISK	Ninguno	A=Estado	Lectura de la pista y el sector en curso y transfieren los datos a la dirección de DMA ya fijada. Un 01 hex deberá volver si existe un error durante la transferencia.
WRITE DISK	Ninguno	A=Estado	Escritura de la pista y el sector en curso de los datos en la dirección de DMA.
SECTOR	BC=Sector lógico	HL=Sector físico	

* Todas las entradas/salidas de consola y los dispositivos deberán realizarse en primer lugar mirando el IOBYTE (0003 hex) para determinar el dispositivo que se ha seleccionado.

Tabla D-1. (Continuación.)

Etiqueta de la tabla de salto	Parámetro(s) de entrada	Parámetro(s) de salida	Aclaración
TRANSLATION	DE=Dirección del mapa de sectores		Una rutina especial utilizada para sistemas que mantienen datos en bloques del orden de 128 bytes. El sector lógico en entrada se ha cambiado para reflejar el sector adecuado del disquete.
LIST STATUS	Ninguno	A=Estado	La rutina debería interrogar al dispositivo adecuado para ver si un carácter está preparado y devolver un 00 hex al registro A si no lo está, o un FF hex si lo está.

Indice

A

ASCII:
Actualización de la hora en ASCII, 243

ASM:
Ensamblador, 201
Manual, 8

AUX:
Dispositivo lógico auxiliar (lectora/perforadora), 65

Anulación (borrado de fichero:
Función 19 del BDOS, 23, 120

Apertura del directorio:
Open_dir, Programa, 406, Explicación, 429

Apertura de fichero:
Función 15 del BDOS, 18, 109
Aproximación a la depuración de Orville Wright: 343

Area de almacenamiento de datos:
Conceptos, 21

Area de programa transitorio:
Encuentro del tamaño disponible, 75

Area reservada:
Conceptos, 21

Argc, argv:
Funciones C para parámetros de orden, 435

Arranque caliente:
Después de error del BDOS, 109
Fijación de la dirección DMA, por defecto, 131
Funciones del BIOS, 165
Iniciado por CONTROL-C, 55
Iniciado por la función de puesta a cero del sistema del BDOS, 82
Iniciado por la pulsación de una tecla, 104
JMP en la situación 0000H, 65
Puesta a cero de discos de sólo lectura, 133
Recarga del CCP, 53
Técnica para evitarlo, 76
Uso de la posición 0000H, 16

Arranque en frío:
Conceptos, 15
Funciones del BIOS, 163

Asignación:
Programa C, asignación de dispositivos lógicos a físicos, 470

Atributos:
En entrada de directorio de ficheros, 31-32

Atributos de fichero: 110
Fijación, 134
(*Véase también* Estado de fichero)

B

BASIC:
Problemas con "trazado" caracteres, 235-236

BDOS:
Acceso al directorio de ficheros, funciones C, 438
Errores detectados, 318
Punto de entrada, en la página base, 69

BIOS:
Bloque de configuración, acceso desde C, 424
Bloque de parámetros de disco, accediendo desde C, 427
Bloqueo/desbloqueo, 167
Componentes, 161
Comprobación en vivo, 396
Conceptos de inicialización, 15
CONIN, entrada de consola, 165
CONOUT, salida de consola, 165
CONST, estado de entrada de consola, 165
Controlador de caracteres, depuración, 379
Cuándo se han de evitar las llamadas directas, 18
Depuración, 377
Depuración de las rutinas de servicio de interrupción, 382
Diferentes tipos de escritura de disco, 170
Ejemplo de programa para BIOS estándar, 174
Encuentro del vector de saltos en RAM, 65
Entrada/salida lógicas, 18
Estructuras de datos mejoradas, 244
Funciones de arranque, 162
Funciones de manejo de errores de hardware, 318
Interfase con otro software, 18
LIST, salida de listado, 166
Listado de BIOS mejorado, 255
LISTST, estado de salida del dispositivo de listado, 171
Llamadas directas al BIOS, ejemplo de programa, 171
Llamadas directas a lectura/escritura de disco desde C, 428
Llamadas directas, ejemplos, 75
Mantenimiento de la fecha actual, 243
Mantenimiento de la hora actual, 243
Mejoras, 225
Mejoras para soportar protocolos diferentes, 236
Memoria-intermedia-grande, HSTBUF, 167
Qué necesita comprobación, 378
Números de salto en LIBRARY.H, 419
Posibilidad de listada de comprobación para depuración, 379
Preparación de una versión especial, 200
Puntos de entrada, 162
READ, lectura de sector, 169
READER, entrada de lectora, 166
Realización de llamadas a C, 424
Retorno a posición inicial de las cabezas de disco, 168
Salida PUNCH (auxiliar), 166

SECTRAN, traducción de sector lógico a físico, 172
Secuencia de operaciones para escritura de sector, 170
SELDSK, selección de disco, 168
SETDMA, fijación de dirección de DMA, 169
SETSEC, fijación de sector, 168
SETTRK, fijación de pista, 168
Soporte de teclas-función, 226
Tabla de dispositivos, acceso desde C, 426
Tabla de teclas-función, accediendo desde C, 425
Utilización de PIP para comprobaciones, 397
Vector de saltos, 18, 65
WRITE, escritura de sector, 170

BIOS estándar:
Ejemplo de programa, 174

Base de comprobación:
Uso para nuevos controladores, 377

Bloque:
Conceptos, 22
Elección de tamaño, 22
En entrada del directorio de ficheros, 31
Número máximo en bloque de parámetros del disco, 41
Prelectura bloque utilizado antes de escritura, 171
Reserva en bloque de parámetros de disco, 42

Bloque de configuración:
Acceso desde C, 424
Códigos variables en LIBRARY.H, 419
Conceptos, 227
Sugerencias para programas de utilidad, 479

Bloque de control de fichero:
Conceptos, 22
Creación de uno de un nombre de fichero ASCII, 111
Definición en LIBRARY.H, 421
Estructura, 48
Usado en apertura de fichero (Open File) del BDOS, 110
Usado para operaciones de fichero aleatorio, 52
Usado para operaciones de fichero secuencial, 51
Usado en peticiones del BDOS, 81

Bloque de control de fichero por defecto:
En página base, 70

Bloque de parámetros del directorio:
Definición en LIBRARY.H, 421

Bloque de parámetros de disco:
Acceso desde C, 427
Adición de información adicional, 48
Apuntador en cabeza de parámetros de disco, 38
Definición en LIBRARY.H, 422
Desplazamiento de bloques, enmascaramiento y máscara de extensión, 40
Detalles, 40
Ejemplo elaborado para disco rígido, 47
Encuentro de la dirección de, 138
Número de entradas del directorio-1, 42
Número máximo de bloque, 41
Número de pistas antes del directorio, 43
Reserva de bloques para directorio de ficheros, 42
Sectores por pista, 40
Tamaño de la memoria intermedia para detección de disquetes cambiados, 43

Bloqueo/desbloqueo:
Conceptos, 44
En el BIOS, 167
Tipos de escritura de disco de BDOS a BIOS, 171

Borrado de caracteres:
Rubout/Del, 58

Borrado de un fichero:
Sólo borrado lógico, 28
ERA, 60

Borrado lógico de ficheros: 28
ERA, 60

Borrado del mapa de disco:
Dm_clr, Programa, 410, Explicación, 433

Búsqueda de fichero:
Srch_file, Programa, 404, Explicación, 439

Búsqueda del siguiente nombre que corresponda:
Función 18 del BDOS, 82, 118
Petición de búsqueda del primero, 115

Búsqueda de primero/siguiente:
Ejemplo de uso juntos, 118

Bv_and:
Vector de bits, AND booleano, Programa, 417, Explicación, 434

Bv_disp:
Vector de bits, visualización, Programa, 417, Explicación, 434

Bv_fill:
Vector de bit, llenado, Programa, 415, Explicación, 434

Bv_nz:
Vector de bits, bit de control distinto a cero, Programa, 416, Explicación, 434

Bv_or:
Vector de bits, OR inclusivo, Programa, 417, Explicación, 434

Bv_set:
Vector de bits, fijación de bit, Programa, 415, Explicación, 434

Bv_test:
Vector de bits, comprobación de bits, Programa, 416, Explicación, 434

C

CARRIAGE RETURN: (retorno de carro):
CONTROL-M, 56

CBIOS.ASM:
Un ingrediente para un nuevo sistema, 201

CCP:
Caracteres de control y sus efectos, 55

Carga de programas, 63
Conceptos de inicialización, 15
Detalles, 53
Dispositivos lógicos, 66
Edición de línea de orden, 55
Ejemplo de volcado de memoria en la página base, 71
ERA, elimina ficheros (barra), 60
Escritura sobre el CCP para ganar memoria, 53
Fijación del FCB por defecto en la página base, 70
Fijación de parámetros de orden en la página base, 70
Funciones, 54
Indicación, 54
Interfase con otro software, 18
Memoria intermedia de DMA por defecto en página base, 71
Modificación de indicación para mostrar el número de usuario, 255
Ordenes incorporadas, 58
Ordenes residentes, 17
Página base, fijación de programa cargado, 201
Recarga en arranque caliente, 53
REN, cambio de nombre de fichero, 61
Retorno sin arranque caliente, 76
SAVE, salvar imagen de memoria en disco, 62
TYPE, visualización de un fichero ASCII, 61
USER, cambio de número de usuario, 62
Vista general, 15

CCPM:
Ejemplo de obtención de número de versión del CP/M, 105

CDISK:
Ejemplo de sistema de puesta a cero de disco, 106

CON:
Consola lógica, 19

CONIN:
Acceso a fecha y hora, 241
Entrada de consola, en el BIOS, 165
Reconocimiento de caracteres de entrada de tecla-función, 239
Uso con entrada forzada, 237

CONOUT:
Procesamiento de secuencias de escape de salida, 240
Salida de consola, en el BIOS, 165
Secuencias de escape para introducir datos y hora, 242

CONST:
Estado de entrada de consola, en el BIOS, 58
Problemas con programas que "tragan" caracteres, 236
Uso de entrada forzada, 237

CONTROL-C:
Usado para abortar después de error del BDOS, 109

CONTROL-P:
Errores generados, 322

CONTROL-Z:
Si no hay ningún controlador de lectora en el BIOS, 86
Usado para indicar final de fichero, 123
Usado para terminar antes del cerrado de fichero (Close File) del BDOS, 113

CP/M:
Creación de un nuevo sistema, 374

CRC:
(*Véase* Control de redundancia cíclica)

CRF:
Ejemplo de escritura aleatoria, 149

Cabeza de parámetros de disco:
Apuntador a vector de situación, 39
Area de trabajo para detección de disquete cambiado, 38
Bloque de parámetros del disco, 38
Desplazamiento de sectores (Sector skewing), 34
Detalles, 34
Memoria intermedia de directorio, 38

Cambio de discos:
Necesidad de forzar la "entrada" en el disco, 107

Cambio se disquete:
Area de trabajo en cabeza de parámetros de disco, 38
Tamaño de la memoria intermedia para detección, en el bloque de parámetros de disco, 43

Cambio de nombre de un fichero:
Función 23 del BDOS, 28, 127

Cambio de número de usuario:
USER, 62

Cancelación de línea de orden:
CONTROL-U, 57

Caracteres de control:
Utilizados en la edición de líneas de orden del CCP, 55

Carga del CP/M:
Vista general, 13

Carga de programas:
Por medio del CCP, 64

Cargador:
Construcción de una nueva versión, 200
Depuración, 375
Ejemplo de programa, 214
Vista general, 10

Cargador en PROM:
Usado para cargar el CP/M, 14

Cerrado de fichero:
Función 16 del BDOS, 113

Código de directorio:
Devuelto por apertura de fichero (Open File), 110
Devuelto por búsqueda del caso siguiente (Next Name Match), 118
Devuelto por búsqueda del primer caso (First Name Match), 114
Devuelto por cerrado de fichero (Close File), del BDOS, 113
Cómo retorna después de cambio de nombre de fichero (Rename File), 127
Cómo retorna después de creación de fichero (Create File), 126
Cómo retorna por llamadas del BDOS, 81

Códigos de función de BDOS: 79
Conceptos de inicialización, 15
Lo que hace el BDOS, 77
Interfase con otro software, 18
Introducción a las llamadas a función, 26
En LIBRARY.H, 419
Nomenclatura, 78
Realización de llamadas en C, 423
Realización de petición de función, 78
Registros convenidos para peticiones de función, 80
Uso de la función 0 después de error de hardware, 322
Uso de la función 0 después de error de impresora, 242
Uso de la posición 0005H, 17-18

Comp_fname:
Comparación de nombres de fichero, Programa, 402, Explicación, 430

Comparación de nombres de fichero:
Comp_fname, Programa, 402, Explicación, 430

Comparación de subcadenas:
Mayúsculas: Usstrcmp, 401
Sstrcmp, 401

Compensación de pistas (Track buffering):
Mejora de entrada/salida de disco, 251

Comunicaciones:
Utilización de lectora/perforadora (auxiliares), 166

Comunicaciones con pregunta por carácter:
Usando entrada/salida directa de consola, 90

Construcción de un fichero indexado:
Utilización de fijación del número de registro aleatorio, 157

Construcción de un nuevo sistema CP/M:
Ejemplo de diálogo de consola, 223
Pasos principales, 199

Construcción del primer sistema CP/M: 152

Contadores de pasos:
Uso de subrutinas de depuración, 347

Control de redundancia cíclica:
Cómo se usa en errores de disco, 326

Controladores caracteres:
Ejemplo de base de comprobación, 380

Controladores de disco:
Depuración, 394

Controladores de disco:
Ejemplo de programa de base de comprobación, 392

Conv_dfname:
Conversión de nombre de fichero de directorio, Programa, 403, Explicación, 432

Conv_fname:
Conversión de nombre de fichero, Programa, 403, Explicación, 439

Conversión de nombre de fichero de directorio:
Conv_dfname, Programa, 403, Explicación, 432

Creación (hacer) de un fichero:
Función 22 del BDOS, 124

Ct_code:
Tabla de códigos, retorno de código, Programa, 413, Explicación, 438

Ct_disps:
Tabla de códigos, visualización de todas las cadenas, Programa, 413, Explicación, 438

Ct_index:
Tabla de códigos, retorno de índice, Programa, 414; Explicación, 438

Ct_init:
Tabla de códigos, inicialización, Programa, 412; Explicación, 437

Ct_parc:
Tabla de código, Código de indicación y retorno, Programa, 412, Explicación, 437

Ct_strc:
Tabla de códigos, obtención de cadena por código, Programa, 414, Explicación, 438

Ct_stri:
Tabla de código, obtención de cadena para índice, Programa, 414, Explicación, 438

Cubo de bits (bit buchet):
Si no se utiliza ninguna unidad de perforación, 87

D

DATE:
Programa C, fija la fecha, 473

DDT:
Herramienta de depuración dinámica, 201, 353
Manual, 8
Orden I usada para construir un nuevo sistema CP/M, 212
Orden R usada para construir un nuevo sistema CP/M, 212
Usado para controlar imágenes del CP/M, 221
Usado para crear la imagen en memoria del CP/M, 207
Usado para depuración de controladores de caracteres, 379
Usado para depurar controladores de disco, 391

DESPOOL:
Uso de entrada LISTST del BIOS, 171

DIR:
Visualización del directorio de ficheros, 59
Dirección de DMA por defecto, 131

DO:
Sugerencias para programas de utilidad, 480

DPB:
(*Véase* Bloque de parámetros de disco)

DPH:
(*Véase* Cabeza de parámetros de disco)

DTR:
PROTOCOL, programa C para fijar protocolos, 465
(*Véase también* Terminal de datos dispuesto)

Densidad simple, una sola cara:
Formato del disquete, 11

Depuración básica para un BIOS: 342
Depurando un nuevo sistema CP/M, 341

Desbordamiento de la memoria:
Depuración del controlador de caracteres, 384

Desplazamiento de bloques:
En bloque de parámetros del disco, 39

Desplazamiento de pistas:
(*Véase* Pistas antes del directorio)

Desplazamiento relativo de página:
Uso para hacer llamadas directas al BIOS, 75

Desplazamiento de sectores:
En la cabeza de parámetros del disco, 34
Efecto en prestaciones, 36
Para la imagen CP/M en el disco, 206

Desplazamiento de sectores (Shewing):
(*Véase* Entrelazamiento de sectores)

Digital Research:
Manuales, 8

Directivas IF/ENDIF:
Utilizado para subrutinas de depuración, 346

Directorio de ficheros:
Acceso directamente a las entradas, 428
Acceso, funciones C, 429
Acceso, por medio del BDOS & funciones C, 438
Bits de estado (atributo), 31
Borrado de ficheros, ERA, 60
Comparación de nombres, funciones C, 430
Concepto, 21
Detalles, 22
Estructura de entrada, 27
Extensión de fichero, 32
Mapa del disco, 32
Nombre y tipo de fichero en entrada, 33
Número de entradas-1, en bloque de parámetros de disco, 42
Número de pistas antes, 43
Número de registro, 32
Número de usuario a la entrada, 27
Proceso, funciones C, 431
Visualización de contenidos, 59

Directorio de lectura/escritura:
Rw_dir, Programa, 408, Explicación, 429

Directorio de sectores de repuesto:
Depuración de controladores de disco, 394

Disco por defecto:
Cambio, 59
En indicación del CCP, 54
En página base, 69

Disco duro:
Consideraciones especiales, 43
División en varios lógicos, 46

Disco de lectura/escritura:
Llamadas directas al BIOS desde C, 428

Disco lógico:
División del disco rígido en varios discos lógicos, 46
Cómo se representa en el bloque de control de fichero, 50
Selección, 108

Disco lógico en curso:
En página base, 69

Disco lleno:
Error devuelto desde escritura secuencial, 124

Disco M:
Usando memoria como un disco ultrarrápido, 252

Disco RAM:
Usando memoria como disco ultrarrápido, 252

Discos de sólo lectura: 132

Dispositivos lógicos:
CON:, 1.ST:, AUX:, RDR:, PUN:, 66

Disquete:
Distribución del disquete estándar CP/M, 45

Disquetes editados:
Ficheros de Digital Research, 8

Distribución del disco:
El CP/M en los disquetes, 206

Distribución de la memoria:
Con CP/M cargado, 16
Por ejemplo BIOS, 207
Para entrada a SYSGEN, 204

Dm_clr:
Borrado de mapa de disco, Programa, 410, Explicación, 433

Dm_disp:
Visualización de mapa de disco, Programa, 410, Explicación, 433

E

Ed:
Editor, manual, 8

ERA:
Anulación (borrado) de ficheros, 60

ERASE:
Programa C, un medio más adecuado para borrar ficheros, 440

Edición de línea de orden:
Por el CCP, 55

Edición de líneas:
Usando lectura de cadenas de consola, 101-102

Ejemplo de programa base de comprobación, 390
TIME, programa C para fijar la hora, 475

Ejemplos de programas:
Ordenación de disquete, 4

Elección de tamaño de bloques: 22

Encuentro (Find):
Programa C, encuentra ficheros perdidos, 447

Enrollamiento de compensador:
Depuración del controlador de caracteres, 386

Enteros cortos:
Cambio menor del lenguaje C, 423

Entrada de directorio: 110
Definición en LIBRARY.H, 422

Entrada en disco:
Usando el sistema de puesta a cero del disco del BDOS, 106-107

Entrada forzada:
Conceptos, 237
Depuración del controlador de caracteres, 386
Sugerencia para programas de utilidad, 479
Entrada de lectora con pregunta por carácter: Problemas y soluciones, 85-86

Entrada/salida:
Entrada/salida ficticia con propósitos de depuración, 351

Entrada/salida de caracteres:
En el BIOS, 165
Interrupciones para entrada, 232
Manejo práctico de errores, 322
Mejoras, 228-230

Entrada/salida de disco:
En el BIOS, 167
Mejoras, 251

Entrada/salida lógica:
Como la permite el BIOS, 18

Entrelazamiento:
(*Véase* Desplazamiento de sectores)

Entrelazamiento de sector:
(*Véase* Desplazamiento de sectores)

Err_dir:
Visualización de error del directorio, Programa, 409, Explicación, 429

Error de construcción:
Entrada/salida de caracteres, manejo, 323

Error de desbordamiento:
Entrada/salida de caracteres, manejo, 323

Error de paridad:
Entrada/salida de caracteres, manejo, 323

Error de tiempo excedido en impresora:
Manejo, 323

Errores:
Ejemplo de rutina de error de impresora, 324
Hardware, análisis, 319
Hardware, corrección, 321
Hardware, estrategia de detección, 318
Hardware, indicación, 319
Manejo de los errores de disco, 326
Manejo práctico de entrada/salida de caracteres, 322
Mensajes de error de disco mejorados, 335
Procesamiento errores hardware, 317

Errores del BDOS:
R/O, 133
Sector dañado, 108, 170
Selección, 108, 168

Errores de disco:
Estrategia, 326

Errores del dispositivo de listado:
Problemas con la función 5 del BDOS, 7, 89

Errores de hardware:
Procesamiento de, 317, Capítulo, 9

Errores de impresora:
Ejemplo de rutina, 324
Uso del temporizador-centinela, 242

Escritura aleatoria:
Usando escritura secuencial, 124

Escritura aleatoria con rellenado de ceros:
Función 40 del BDOS, 158
Escritura 34 del BDOS, 147

Escritura de un byte en la consola:
Función 2 del BDOS, 83

Escritura de byte por dispositivo de listado:
Función 5 del BDOS, 88

Escritura de byte por perforadora:
Función 4 del BDOS, 86

Escritura sobre el CCP:
Para ganar memoria, 57

Escritura diferida:
En conjunción con memoria intermedia de pista, 251

Escritura de disco (por medio del BIOS):
Wrt_disk, Programa, 405, Explicación, 430

Escritura secuencial:
Función 21 del BDOS, 122

Estado del archivo:
En entrada de directorio de ficheros, 31

Estado de consola:
Depuración de controlador de caracteres, 387

Estructura:
Del CP/M, 7

Estructura de fichero COM: 211

Estructura de fichero HEX: 211

Etx/Ack:
Depuración de controladores de caracteres, 384, 388-389
Explicación de protocolo, 236-237
Protocolo, programa C para fijar protocolos, 465

Examen de cadena:
Strscn, 400

Examen de cadena de mayúsculas:
Ustrscn, 400

Expansión de tab:
Soportada por escritura de byte de consola, 84
Usando visualización de cadena terminada en $, 99

Extensión:
En entrada de directorio de ficheros, 32
De ficheros, conceptos, 22

Extensión de fichero:
Apertura de extensión 0 para entrada/salida aleatoria, 146-148
Conceptos, 22
En entrada de directorio de ficheros, 32
Manipulación para realizar entrada/salida aleatoria, 122-124

F

FCB:
FCB, por defecto en página base, 70
(*Véase también* Bloque de control de fichero)

FDOS:
Término raramente usado para BDOS/CCP combinados

FUNKEY:
Programa C, fija las teclas-función, 476

Fecha:
Lectura de la fecha desde el controlador de consola, 241
Mantenimiento de la fecha actual en el BIOS, 243

Ficheros:
Creación, secuencia de operaciones, 25-26
Visualización de directorio, DIR, 59

Ficheros aleatorios:
Conceptos, 52
Creación de un fichero vacío, 158
Problema de ficheros dispersos, 52
Tamaño virtual, 156

Ficheros borrados:
Recuperación, 32

Ficheros cambiados:
Bit de estado de fichero en entrada del directorio de ficheros, 32

Ficheros COM:
Cargados por el CCP, 54

Ficheros distribuidor al azar:
Problema, 52

Ficheros de sólo lectura:
Bit de atributos, 134

Ficheros secuenciales:
Conceptos, 51

Ficheros sistema:
Bit de atributo, 134
Bit de estado de fichero en la entrada del directorio de ficheros, 31
No visualizado por DIR, 59

Ficheros públicos:
Piezas para crear éstos, posibilidad, 255
Sugerencia para programas de utilidad, 479

Fijación de atributos de fichero:
Función 30 del BDOS, 36, 134

Fijación de bloque de control de búsqueda:
Setscb, Programa, 409, Explicación, 431

Fijación de la dirección DMA (lectura/escritura):
Función 26 del BDOS, 131
Necesario para búsqueda del primer nombre coincidente, 115

Fijación de disco lógico a sólo lectura: Función 28 del BDOS, 132

Fijación del IOBYTE:
Función 8 del BDOS, 96

Fijación del número de registro aleatorio:
Función 36 del BDOS, 156

Fijación/obtención del número de usuario:
Función 32 del BDOS, 144

Fijación de parámetros de disco para rd/wrt_disk:
Set_disk, Programa, 406, Explicación, 429

Filecopy:
Sugerencia para programas de utilidad, 457

Final de fichero:
Detección usando lectura secuencial, 123

Final de línea físico:
CONTROL-E, 55

Formato de disquete:
Conceptos, 11

Funciones BDOS:
0, Puesta a cero del sistema, 81
1, Lectura de un byte de la consola, 82
2, Escritura de un byte en la consola, 83
3, Lectura de byte de lectora, 84
4, Escritura de byte por perforadora, 86
5, Escritura de byte por dispositivo de listado, 88
6, E/S directa de la consola, 89
7, Obtención del montaje IOBYTE, 95
8, Fijación del IOBYTE, 96
9, Visualización de cadena terminada en $, 99
10, Lectura de cadena de la consola, 101
11, Lectura de estado de la consola, 104
12, Obtención del número de versión de CP/M, 105

13, Puesta a cero del sistema de disco, 106
14, Selección del disco lógico, 108
15, Apertura de fichero, 109
16, Cerrado de fichero, 113
17, Búsqueda del primer nombre que corresponda, 114
18, Búsqueda del siguiente nombre que corresponda, 118
19, Borrado de fichero, 120
20, Lectura secuencial, 121
21, Escritura secuencial, 122
22, Creación de un fichero, 124
23, Cambio de nombre de un fichero, 127
24, Obtención discos activos, 129
25, Obtención del disco actual por defecto, 130
26, Fijación de la dirección DMA, 131
27, Obtención del vector de situación, 132
28, Situación del disco, lógico en modo de sólo lectura, 132
29, Obtención de los discos en estado de sólo lectura, 133
30, Fijación de atributos de fichero, 134
31, Obtención de la dirección del bloque de parámetros de disco, 138
32, Fijación/obtención del número de usuario, 144
33, Lectura aleatoria, 145
34, Escritura aleatoria, 147
35, Obtención del tamaño de fichero, 155
36, Fijación del número de registro aleatorio, 156
37, Puesta a cero del controlador de disco lógico, 157
40, Escritura aleatoria con rellenado de ceros, 158

G

GETC:
Ejemplo de lectura secuencial, 122

GETDPB:
Ejemplo de obtención de dirección de bloque de parámetros de disco, 139

GFA:
Ejemplo de obtención de atributos de fichero, 135

GNFC:
Ejemplo de búsqueda primer caso, caso siguiente, 115

Get_cba:
Obtención de dirección del bloque de configuración, 400

Get_dpb:
Obtención de dirección del bloque de parámetros, Programa, 411

Get_nde:
Obtención de la próxima entrada del directorio, Programa, 406, Explicación, 429

Get_nfn:
Obtención del próximo nombre de fichero, Programa, 404, Explicación, 439

H

HOME:
Inicialización de cabezas de disco, en el BIOS, 168

HSTBUF:
En el BIOS, 167

Heath/Zenith:
Versión especial del CP/M, 65

I

IOBYTE:
Equivalencias para campos de bits, 96
Estructura, 66
Uso con entrada/salida directa de consola para comunicaciones, 91
Uso para examinar línea de comunicación, 86

Imagen de memoria:
Comprobación de un sistema nuevo, 221
De un nuevo sistema CP/M, 202

Impresoras paralelo:
Procesamiento de errores, 324

Indicación:
Desde el CCP, 54

Inicialización de subrutinas de depuración: 346

Interacciones:
Entre CCP, BDOS y BIOS, 19

Interrupciones:
Arquitectura, 233
Depuración de las rutinas de servicio, 352
Memorias intermedias circulares, 235
Procesamiento de exceso de datos en la memoria intermedia, 237
Uso para controladores de entrada de caracteres, 232

J

Johnson-Laird Inc:
Pedidos de disquetes, 5

L

LIBRARY.C:
Biblioteca de funciones de utilidad, 400

LIBRARY.H:
Cabeza para funciones de LIBRARY.C, 418

LIST:
Salida de listado, en el BIOS, 166

LISTST:
Estado de salida de dispositivo de listado, en el BIOS, 171

LST:
Dispositivo lógico de listado, 66

Lectura aleatoria:
Función 33 del BDOS, 145

Lectura aleatoria:
Usando lectura secuencial, 123

Lectura de un byte de la consola:
Función 1 del BDOS, 82

Lectura de byte de lectora:
Función 3 del BDOS, 84

Lectura de cadena de la consola:
Función 10 del BDOS, 101

Lectura de disco (por medio del BIOS):
Rd_disk, Programa, 405, Explicación, 429

Lectura de estado de la consola:
Función 11 del BDOS, 104

Lectura secuencial:
Función 20 del BDOS, 121

Lenguaje C:
Manuales de referencia, 4
Utilización para programas de utilidad, 399

Letras minúsculas en nombre de fichero: 126

Línea de orden:
Borrado del último carácter mecanografiado, 57
Eliminación, CONTROL-U, 57
Repetición, CONTROL-R, 57

Línea de orden Undo:
CONTROL-U, 57

Lista de control de depuración:
Controladores de disco, 391
Reloj de tiempo real, 389
Rutinas de servicio de interrupción, 385
Rutinas de servicio de no interrupción, 385
Salida de caracteres, 388

Loadsys:
Sugerencias para programas de utilidad, 479

Longitudes de registro variables:
Procesamiento en ficheros aleatorios, 147, 148

LL

Llamadas directas al BIOS:
Cuando hay que evitarlas, 18
Ejemplo de programa, 171
Ejemplos, 76

M

MAC:
Macroensamblador, 201

MAKE:
Programa C, hace ficheros visibles/invisibles, 458

M80:
Macroensamblador, 201

MOVCPM:
En conjunción con piezas del CP/M, 254
Reubicación del CCP y el BIOS, 219
Uso en la construcción de un nuevo sistema CP/M, 198

MOVE:
Programa C, traslada ficheros entre números de usuario, 454

MSGOUT:
Ejemplo de escritura de byte consola, 84

MSGOUTI:
Ejemplo de escritura de byte consola, 85

Manejo de sectores dañados: 327
En el BIOS, 169
Sugerencia para programas de utilidad, 457, 479

Manuales:
De Digital Research, 8

Mapa de bits:
(*Véase* Vector de situación)

Mapa de disco:
En entrada de directorio de ficheros, 32

Mapa de disco:
Cómo se usa en las funciones C, 432

Máscara de bloques:
En bloque de parámetros del disco, 39

Máscara de extensión:
En bloque de parámetros del disco, 39

Mecanografiado en avance (Type-ahead):
Conceptos, 234
Procesamiento de desbordamiento de memoria intermedia, 237

Memoria:
Encuentro del tamaño del área disponible para programas, 75
Usada como un disco ultrarrápido, 252
Uso de memoria "desconocida" para memoria intermedia, 234
Visualización en subrutinas de desinsectación, 347

Memoria intermedia circular:
Para entrada de control de interrupciones, 234
Estructura de la tabla de dispositivos, 245

Memoria intermedia de disco:
En cabeza de parámetros de disco, 38

Memoria intermedia de DMA:
Por defecto, en página base, 70

Memoria intermedia de DMA por defecto:
En página base, 70

Memoria intermedia de flujo:
Antes de cerrado de fichero (Close File) del BDOS, 113

Memoria intermedia grande (host buffer):
En el BIOS, 167

Mensajes:
Como ayuda a depuración, 350

Mensajes de error:
Depuración de controladores de disco, 395, Capítulo, 12

N

Nombre/tipo de fichero:
En entrada de directorio de fichero, 28

Nombres de fichero ambiguos:
Conceptos y restricciones, 29
Ejemplo de procesamiento, 430
Evitar el cambio de nombre de fichero, 127
Sugerencia para programas de utilidad, 457
Utilizado en apertura de ficheros BDOS, 110
Utilizado en búsqueda del primer nombre coincidente, 114
Utilizado en DIR, 59
Utilizado en ERA, 60

Notación:
Por ejemplo diálogo de consola, 3

Nuevo nombre de un fichero:
REN, 61

Número de entradas al directorio de ficheros:
En el bloque de parámetros del disco, 42

Número máximo de bloque:
En el bloque de parámetros del disco, 41

Número de registro:
En la entrada del directorio de ficheros, 32
Manipulación para terminar la entrada/salida aleatoria, 123, 124

Número de registro aleatorio:
En el FCB, fijado para escritura aleatoria, 146
En el FCB, fijado para lectura aleatoria, 146

Número de registro en curso:
En FCB, invariable para escritura aleatoria, 146
En FCB, invariable para lectura aleatoria, 146

Número de usuario:
Cambio bajo control de programa, 144
Cambio usando USER, 62
En la entrada del directorio de ficheros, 27
Modificación para hacerlo aparecer en la indicación del CCP, 255
En la página base, 69
Sugerencia para programas de utilidad, 457
Visualización, 63

Número de usuario en curso:
En página base, 69
Visualización, 63

O

OM:
Ejemplo de visualización de cadena terminada en $, 100

OPENF:
Ejemplo de apertura de fichero, 111

Obtención de la configuración del IOBYTE:
Función 7 del BDOS, 95

Obtención de dirección del bloque de configuración:
Get_cba, 400

Obtención de la dirección del bloque de parámetros de disco:
Función 31 del BDOS, 138

Obtención de la dirección del bloque de parámetros de disco:
Get_dpb, Programa, 411

Obtención del disco actual por defecto:
Función 25 del BDOS, 130

Obtención de los discos en estado de sólo lectura:
Función 29 del BDOS, 133

Obtención del número de versión del CP/M:
Función 12 del BDOS, 105

Obtención de próxima entrada del directorio:
Get_nde, Programa, 406, Explicación, 429

Obtención de próximo nombre de fichero:
Get_nfn, Programa, 404, Explicación, 439

Obtención del tamaño de fichero:
Función 35 del BDOS, 155

Obtención del vector de situación:
Función 27 del BDOS, 132

Open_dir:
Apertura de directorio, Programa, 406, Explicación, 429

Ordenes incorporadas:
En el CCP, 54

Ordenes residentes del CCP: 17

Organizaciones de fichero:
Conceptos, 49

P

PIP:
Usado para comprobar un BIOS nuevo, 397

PROM fantasma (Shadow PROM):
Usada para cargar el CP/M, 13

PROTOCOL:
Programa C, fija protocolos de línea serie, 465

PUN:
Perforadora lógica, 66

PUNCH:
Salida de perforación (auxiliar), en el BIOS, 166

PUTC:
Ejemplo de escritura secuencial, 125

PUTCPM:
Ejemplo de programa, 208
Escritura de una utilidad, 206

Página base:
Ejemplo de volcado de memoria, 71
Fijación por el CCP para programa cargado, 64
Número de usuario en curso, 69

Parámetros:
Programa ejemplo para procesar parámetros de orden, 73

Personalización:
De CP/M, vista general, 10

Petición de envío:
Explicación del protocolo RTS, 236-237

Pista:
Llenado con configuración conocida, 346

Pista excedida:
En rutina de servicio de interrupción, 383

Pistas antes del directorio:
En el bloque de parámetros de disco, 43

Posición 0000H:
Uso para arranque caliente, 16

Posición 0005H:
Ejemplos sencillos de uso, 26
Uso para llamadas a función del BDOS, 17

Prestaciones:
Efecto del desplazamiento de sectores, 35

Principio de la RAM:
Encontrar, por medio de página base, 70

Procesador de órdenes de consola:
(*Véase* CCP)

Programas C:
ASSIGN, asignación de dispositivos lógicos a físicos, 469
DATE, fijación de la fecha, 473
ERASE, un modo mejor de eliminar ficheros, 440
FIND, encuentra ficheros perdidos, 447
FUNKEY, fija teclas-función, 476
MAKE, hace los ficheros visibles/invisibles, 458
MOVE, traslada ficheros entre números de usuario, 454
PROTOCOL, fija protocolos de línea serie, 465
SPACE, muestra espacios del disco utilizados/libres, 450
SPEED, fija velocidad de transmisión (baud rate), 462
TIME, fija la hora, 473
UNERASE, recupera ficheros eliminados, 442

Programa de depuración incorporado: 343

Programas de utilidad: 399

Protección de ficheros:
Caracteres especiales en el nombre de fichero, 126

Protección de ficheros:
Sugerencia para programas de utilidad, 457

Propietario:
Sugerencia para programas de utilidad, 457

Protegido/no protegido:
Sugerencia para programas de utilidad, 457

Protocolo:
Definiciones en LIBRARY.H, 420
Soporte en el BIOS mejorado, 236
Soporte por medio de la tabla de dispositivos, 245
Xon/Xoff, usado por TYPE, 62
(*Véase también* Terminal de datos preparado; Petición de envío; Xon/Xoff; Etx/Ack)

Público/privado:
Sugerencia para programas de utilidad, 479

Puesta a cero:
Señal usada para empezar a cargar el CP/M, 13

Puesta a cero del controlador de disco lógico:
Función 37 del BDOS, 157

Puesta a cero de disco único: 157

Puesta a cero del hardware:
Depuración controlador de caracteres, 385

Puesta a cero del sistema:
Función 0 del BDOS, 81

R

RAM disponible:
Búsqueda de cantidad disponible, 75

RCS:
Ejemplo de entrada salida directa de consola, 91

RDR:
Lectora lógica, 66

READ:
Lectura de un sector, en el BIOS, 169

READER:
Entrada de lectora, en el BIOS, 166

REN:
Cambio de nombre de fichero, 61

RF:
Ejemplo de cambio de nombre de fichero, 128

RL$RDR:
Ejemplo de lectura de byte de lectora, 87

RMAC:
Macroensamblador, reubicable, 201

RO:
Ejemplo de entrada/salida de fichero aleatorio, 150

RSA:
Ejemplo de lectura de cadena de consola, 102

RST 7:
Uso para depuración de controladores, 381

RTS:
Protocolo, programa C para fijar protocolos, 465
(*Véase también* Umbral de memoria intermedia; Petición de envío)

Rd_disk:
Lectura de disco (por medio del BIOS), Programa, 405, Explicación, 429

Reclain preservación de sectores dañados:
Sugerencia para programas de utilidad, 457

Recuperación de registros:
En rutina de servicio de interrupción, 381

Redireccionamiento de entrada:
Depuración de controlador de caracteres, 385

Redireccionamiento de entrada/salida:
Asignación, programa C para asignar dispositivos físicos, 470
Conceptos, 231
Estructura IOBYTE, 66-67

Reflejo de caracteres de teclado:
Lectura de un byte de la consola, 82

Reflejo en impresora:
CONTROL-P, 56

Registros:
Visualización en subrutinas de depuración, 347

Registros del CP/M como sectores de 128 bytes: 81

Registros lógicos:
Conceptos, 49

«Registros» de 128 bytes del CP/M: 49

Reloj de tiempo real:
Depuración, 389

Remiendo del CP/M:
Técnicas generales, 254

Repetición de línea de orden:
CONTROL-R, 57

Repuesto:
Sugerencia para programas de utilidad, 479

Repuestos:
Uso de los sectores de repuesto del disco, 327-328

Retroceso:
CONTROL-H, 56

Rutinas de servicio de interrupción:
Depuración de lista de control, 382

Rw_dir:
Lectura/escritura de directorio, Programa, 408, Explicación, 429

S

SAVE:
Salva imagen de memoria en fichero de disco, 62
Uso en la construcción de un nuevo sistema CP/M, 211

SECTRAN:
Sector lógico a físico, en el BIOS, 172

SELDSK:
Depuración de controladores de disco, 394
Selección de disco, en el BIOS, 168

SETDMA:
Fija la dirección de DMA, en el BIOS, 169

SETSEC:
Fija el sector, en el BIOS, 168

SETTRK:
Fijación de pista, en el BIOS, 168

SETTRK/SETSEC:
Depuración de controladores de disco, 394

SID:
Herramienta de depuración, 354

SFA:
Ejemplo de fijación de atributos de fichero, 135

SPACE:
Programa C, muestra el espacio usado/libre del disco, 450

SPEED:
Programa C, fija la velocidad de transmisión (baud rate), 462

STAT:
Uso para visualización de números de usuario actuales, 63

SYSGEN:
Escritura de un sistema nuevo en el disco, 203
Generador de sistema, 202

Salida de consola:
Desde las subrutinas de depuración, 346
Pausa temporal, CONTROL-S, 57

Salida de consola a impresora:
CONTROL-P, 56

Salida de depuración:
Control cuando ocurre, 346

Salto:
Salto de sectores dañados en el disco, 328

Salto de línea:
CONTROL-J, 56

Salto de sector:
Conceptos, 328

Salvado de memoria en el disco:
SAVE, 62

Savesys:
Sugerencia para programas de utilidad, 479

Sector de repuesto:
Conceptos, 328

Sectores:
Uso en bloques, 22

Sectores físicos:
Relativos, en un disco duro, 45

Sectores físicos relativos:
En un disco duro, 45

Sectores lógicos a físicos: 34
SECTRAN, en el BIOS, 172

Sectores por pista:
En el bloque de parámetros de disco, 40

Secuencias de escape:
Entrada, depuración de controlador de caracteres, 387
Teclas-función, depuración controlador de caracteres, 387
Procesamiento de secuencias de salida, 240
Reconocimiento de secuencias de tecla-función, 240
Soporte por medio de la tabla de dispositivos, 245
Sugerencia para programas de utilidad, 479

Secuencia de escape de salida:
Depuración del controlador de salida de carácter, 389

Secuencias de escape estándar ANSI:
Soporte por medio del BIOS, 238

Selección del disco lógico:
Función 14 del BDOS, 108

Set_disk:
Fijación parámetros de disco para rd/wrt_disk, Programa, 406, Explicación, 430

Setscb:
Fijación de bloque de control de búsqueda, Programa, 409, Explicación, 430

Setterm:
Sugerencia para programas de utilidad, 479

Sistema de ficheros:
Conceptos, 21

Sistema de ficheros del CP/M:
Conceptos, 21

Sistema operativo básico de disco:
(*Véase* BDOS)

Sistema de puesta a cero del disco:
Función 13 del BDOS, 106

Sólo lectura:
Bit de estado de fichero en la entrada del directorio de ficheros, 31
Fijación automática después de cambiar los disquetes, 38

Srch_file:
Busca fichero, Programa, 404, Explicación, 439

Sstrcmp:
Compara subcadenas, 401

Strscn:
Examen de cadena, 400

Subrutina:
CCPM, Controla si es la versión 2 del CP/M, 106
CDISK, Cambia el disco, 107
CRF, Crea fichero aleatorio, 149
DB$Blank, Visualiza un espacio, 368
DB$CAH, Convierte A en ASCII Hex., 367
DB$CRLF, Compone retorno de carro, salto de línea, 368
DB$Colon, Visualiza una coma, 368
DB$Conin, Depuración de entrada de consola, 360
DB$Conout, Depuración de salida de consola, 360
DB$DAH, Visualiza A en Hex., 367
DB$DHLH, Visualiza en Hex., 367
DB$Display$CALLA, Visualiza dirección de llamada, 367
DB$Display, Visualización principal de depuración, 362
DB$GHV, Obtención de valor Hex., 372
DB$Init, Inicialización de depuración, 359
DB$Input, Entrada puerto de depuración, 370
DB$MEMORY, Depura visualización de memoria/registros, 348
DB$MSG, Visualización de mensaje, 369
DB$MSGI, Visualización de mensaje (en línea), 369
DB$Off, Desconecta la salida de depuración, 361
DB$On, Conecta la salida de depuración, 361
DB$Output, Salida puerto de depuración, 371
DB$Pass, Decrementa contador de pasos, 361
DBSetPass, Fija contador de pasos, 361
DIVHL, Divide HL por DE, 142
FOLD, Transforma las minúsculas en mayúsculas, 104
FSCMP, Comparación de cadenas transformadas en mayúsculas, 104
GAB, Obtención de bloque dado el sector y pista, 141
GDTAS, Obtención de pista/sector de directorio, 140
GETC, obtención de carácter de un fichero secuencial, 122
GETDPB, Obtención de dirección del bloque de parámetros de disco, 139
GFA, Obtención de atributos de fichero, 135
GMTAS, Obtención de pista/sector máximo, 140
GNF, Obtención fichero siguiente que coincide con nombre ambiguo, 115
GNTAS, Obtención de pista/sector siguientes, 141
GTAS, Obtención de pista/sector del bloque número, 139
MSGOUT, Salida de mensaje, 84
MSGOUTI, Salida de mensaje en línea, 85
MULHL, Multiplicación de HL por DE, 142
OM, Mensaje de salida seleccionado por el registro A, 100
OPENF, Apertura de fichero dado un nombre de fichero ASCII, 111
PUTC, Inclusión de carácter en un fichero secuencial, 125
RCS, Lectura de cadena de consola, 91
RF, Cambio de nombre de fichero, 128
RL$RDR, Lectura de línea desde la lectora, 87
RO, Entrada/salida de fichero aleatorio (registros no 128 bytes), 150
RSA, Retorno dirección del subprocesador, 103
SDLR, Desplazamiento de DE,HL un bit a la derecha, 155
SFA, Fijación de atributos de fichero, 135
SHLR, Desplazamiento de HL un bit a la derecha, 143
SUBHL, Sustracción de DE de HL, 143
TERM, Emulador de terminal, 98
TOUPPER, Transformación de minúsculas en mayúsculas, 94
WL$LST, Escritura de línea en el dispositivo de listado, 90
WL$PUN, Escritura de línea en perforadora, 88
Subrutinas de depuración, 344
Filosofía general de diseño, 344

T

TERM:
Ejemplo de fijación/obtención del IOBYTE, 98

TIME:
Programa C, fija la hora, 475

TYPE:
Visualiza un fichero ASCII, 61

Tab:
Interacción de caracteres tab y secuencias de escape, 241

Tabla de códigos:
Definición de estructura en LIBRARY.H, 422
Indicación y devolución de código, ct_parc, Programa, 412, Explicación, 437
Inicialización, ct_init, Programa, 412, Explicación, 437
Obtención de cadena por código, ct_strc, Programa, 414, Explicación, 438
Obtención de cadena por índice, ct_stri, Programa, 414, Explicación, 438
Retorno de código, ct_code, Programa, 413, Explicación, 438
Retorno de índice, ct_index, Programa, 414, Explicación, 438
Utilizado para parámetros de orden, 435-436
Visualización de todas las cadenas, ct_disps, Programa, 413, Explicación, 438

Tablas de definición de disco:
Conceptos, 22
Detalles, 33

Tabla de dispositivos:
Acceso desde C, 426
Estructura, 244
Visualización para depuración, 381

Tabla de teclas-función:
Acceso desde C, 425

Tamaño de sector:
Grande (host) en el BIOS, 167

Tamaño del sector grande (host sector size):
En el BIOS, 167

Teclas-función:
Comprobación en un BIOS vivo, 398
Estructura en LIBRARY.H, 420
Soporte con BIOS mejorado, 237

Temporizador-centinela:
Conceptos, 243
Depuración del reloj de tiempo real, 391
Uso para detectar errores de impresión, 242

Terminación del programa:
Regreso al CP/M, 75

Terminación de programas regreso al CCP: 53

Terminal de datos dispuesto:
Explicación del protocolo DTR, 236

Tiempo (Time):
Lectura de la hora desde el controlador de consola, 241
Mantenimiento de la hora actual en el BIOS, 243
Visualuzación correcta durante la depuración, 394

Tipo de fichero:
Acuerdos para tipos, 30

Títulos:
Para subrutinas de depuración, 344

U

UNERASE:
Programa C, recupera ficheros borrados, 442

Umbrales de la memoria intermedia:
Depuración del controlador de caracteres, 386

Unidad por defecto en curso: 108

Ustrcmp:
Examen de cadena de mayúsculas, 400

Usstrcmp:
Comparación de subcadena de mayúsculas, 401

V

Vector de apertura:
Véase Función 24 del BDOS, 129

Vector de bits:
AND booleana, bv_and, Programa, 417, Explicación, 434
Bit de control, bv_test, Programa, 416, Explicación, 434
Bit de control distinto a cero, Programa, 416, Explicación, 434
Cómo se usa en las funciones C, 431
Definición de estructura en LIBRARY.H, 423
Fijación de bit, bv_set, Programa, 415, Explicación, 434
Llenado, bv_fill, Programa, 415, Explicación, 434
OR inclusivo, bv_or, Programa, 417, Explicación, 434
Realización bv_make, Programa, 415, Explicación, 433
Visualización bv_disp, Programa, 417, Explicación, 434

Vector de saltos:
Uso para introducir el BIOS, 18

Vector de situación:
Ajuntados en la cabeza de parámetros de disco, 39
Obtención de dirección de, 132

Velocidad de transmisión (baud rate):
Velocidad, programa C para fijar velocidad de transmisión, 462

Versión especial del CP/M:
Heathkit/Zenith, 65

Visualización de cadena terminada en $:
Función 9 del BDOS, 99

Visualización de errores de directorio:
Err_dir, Programa, 409; Explicación, 429

Visualización de un fichero ASCII:
TYPE, 61

Visualización del mapa de disco:
Dm_disp, Programa, 410, Explicación, 433

Visualización del número de usuario en curso: 63

Volcado de memoria:
Página base, 71

W

WL$LST:
Ejemplo de byte de escritura de listado, 90

WL$PUN:
Ejemplo de byte de escritura en perforadora, 88

WRITE:
Escritura de sector, en el BIOS, 170

Wrt_disk:
Escritura de disco (por medio del BIOS), Programa, 405, Explicación, 429

X

Xoff:
CONTROL-S, 57

Xon:
CONTROL-Q, 57

Xon/Xoff:
Depuración controlador de caracteres, 384, 388
Explicación del protocolo, 260
PROTOCOL, programa C para fijar protocolos, 465
Soportado por lectura de byte de consola, 82
Uso por TYPE, 62

Z

ZSID:
Depurado interactivo simbólico, 280, 201, 374

Zona de parámetros de orden:
Ejemplo de programa para procesamiento de parámetros, 73
Entrada al CCP, 54
En página base, 70
Procesamiento, funciones C, 435
Tablas de código, funciones C, 435

OTRAS OBRAS DE INTERES PUBLICADAS POR OSBORNE/McGRAW-HILL

BERMAN: *Explorando el BASIC en el COMMODORE 64*
CASTLEWITZ: *Introducción al VisiCalc.*
DITLEA: *Guía de Software para macrocomputadoras.*
ERICKSON: *Telecomunicaciones con el C-64.*
ERICKSON: *Telecomunicaciones con el MACINTOSH.*
ETTLIN: *Manual de MBASIC.*
ETTLIN: *Introducción al Wordstar.*
FIEKD: *Aplique el MacWhite y MacPaint.*
FLAST: *1-2-3 RUN. 41 aplicaciones prácticas Lotus 1-2-3.*
FLAST: *54 aplicaciones del VisiCalc.*
FOX: *BASIC básico. Guía para principiantes.*
HEILBORN: *Commodore 64. Guía del usuario.*
HEILBORN: *Programas para ciencias e ingeniería.* Edición Apple II.
HEILBORN: *VIC 20. Guía del usuario.*
HOFFMAN: *Sistema operativo MS-DOS. Guía del usuario.*
HOFFMAN: *MSX. Guía del usuario.*
HOGAN: *Sistema operativo CP/M. Guía del usuario* (2.ª ed.).
JEFFRIES: *Commodore 64. Pasatiempos y juegos.*
KRUGLINSKY: *Guía a las comunicaciones del IBM/PC.*
MOTTOLA: *Programación en lenguaje ensamblador para el Apple II.*
OSBORNE: *Guía del comprador de sistemas de gestión.*
OSBORNE: *Guía del ordenador personal PET/CBM.*
POOLE: *Algunos programas de uso común en BASIC.*
POOLE: *Algunos programas de uso común en BASIC.* Edición Apple II.
POOLE: *Algunos programas de uso común en BASIC.* Edición Atari.
POOLE: *Algunos programas de uso común en BASIC.* Edición IBM.
POOLE: *Algunos programas de uso común en BASIC.* Edición PET/CBM.
POOLE: *Algunos programas de uso común en BASIC.* Edición TRS-80.
POOLE: *Algunos programas de uso común en Pascal.*
POOLE: *Apple II. Guía del usuario.*
POOLE: *Programas prácticos en BASIC.*
POOLE: *Programas prácticos en BASIC.* Edición Apple II.
POOLE: *Programas prácticos en BASIC.* Edición IBM.
POOLE: *Programas prácticos en BASIC.* Edición TRS-80.
POOLE: *Programas prácticos en Pascal.*
SAND: *Pascal avanzado. Técnicas de programación.*
SACHS: *El IBM/PC.*
STEWART: *Juegos y programas educativos para C-64.*
STEWART: *Juegos y programas educativos para Apple.*
THOMAS: *Sistema operativo UNIX. Guía del usuario.*
TOWNSEND: *Aplique el BASE II.*
WAITE: *Introducción al procesamiento de palabras.*

DISCOGUIAS PUBLICADOS POR OSBORNE/McGRAW-HILL

GIFFOD: *Discoguía para Apple II.*
INGRAHAM: *Discoguía para CP/M.*
JOYANES: *Discoguía para ZX SPECTRUM (TS2068).*
TAYLOR: *Discoguía para Atari 400/800.*
WILSON: *Discoguía para IBM/PC.*
WILSON: *Discoguía para VisiCalc.*

OTRAS OBRAS DE INTERES PUBLICADAS POR BYTE-BOOKS/McGRAW-HILL

ABELSON: *LOGO para Apple II.*
ABELSON: *Apple Logo.*
BOWLES: *Introducción al UCSD Pascal.*
CIARCIA: *Construya una computadora basada en el Z-80 (Guía de diseño y funcionamiento).*
CURTIS: *WORDSTAR en el IBM/PC.*
DUFF: *Introducción al MACINTOSH.*
GABY: *Gosubs. 100 subrutinas de uso común para el ZX-81 (TS 1000).*
KAMINS: *Usted y la microcomputadora. (Una introducción humanizada a la microinformática).*
KOLVE: *Guía para seleccionar y adquirir su microcomputador.*
LEWART: *Programas de ciencias e ingeniería para microcomputadoras Sinclair ZX-81 compatibles con el ZX Spectrum.*
MORGAN: *Introducción al microprocesador 8086/8088 (16 bit).*
MULLISH: *Sinclair ZX-81 (TS 1000). Guía del usuario.*
MULLISH: *Applesoft BASIC. Guía para principiantes.*
PECKHAM: *BASIC para Apple II. Manual práctico.*
PECKHAM: *BASIC para Commodore 64. Manual práctico.*
PECKHAM: *BASIC para IBM. Manual práctico.*
PECKHAM: *BASIC para TRS-80 color. Manual práctico.*
SIKONOWIZ: *Introducción al IBM/PC.*
SKIER: *Programación en lenguaje ensamblador para VIC-20 y Commodore 64.*
WATT: *Aprendiendo con LOGO.*
WATT: *Aprendiendo con Commodore LOGO.*

OTRAS OBRAS DE INTERES PUBLICADAS POR McGRAW-HILL

ADAMIS: *Diccionario BASIC.*
ADAMIS: *Diccionario BASIC del IBM/PC.*
ADAMIS: *Fórmulas y programas usuales en BASIC.*
ADAMIS: *Iniciación al BASIC del IBM/PC.*
ADAMIS: *MACINTOSH. Aplicaciones de Multiplan y MacPaint.*
BISHOP: *ZX Spectrum (TS 2068). Teoría y proyectos de interfase.*
BUFFINGTON: *Su primera computadora: cómo comprarla y utilizarla.*
GOSLING: *Programación estructurada.*
HURLEY: *Introducción a la programación ZX-81 (TS 1000).*
HURLEY: *ZX Spectrum (TS 2068). Introducción al procesamiento de textos.*
HURLEY: *ZX Spectrum (TS 2068). Programación para jóvenes programadores.*
NICHOLLS: *ZX Spectrum (TS 2068). Programación de juegos en lenguaje ensamblador.*
PHILLIPS: *Programando el Dragón. Juegos y gráficos.*
STREET: *ZX Spectrum (TS 2068). Técnicas de procesamiento de la información.*
WILLIAMS: *ZX Spectrum (TS 2068). Diseño y programación de juegos.*
WOODS: *ZX Spectrum (TS 2068). Programación en lenguaje ensamblador.*

OTRAS OBRAS DE INTERES PUBLICADAS POR OSBORNE/McGRAW-HILL

CASTLEWITZ: Introducción al Visicalc.
ERICKSON: Telecomunicaciones con el C-64.
ERICKSON: Telecomunicaciones con el Macintosh.
ETTLIN: Manual de MBASIC.
ETTLIN: Introducción al WORDSTAR.
FLAST: 54 aplicaciones del Visicalc.
FOX: BASIC básico. Guía para principiantes.
FIELD: Aplique MacWrite y MacPaint.
GRAHAM: IBM/PC. Guía del usuario.
HEILBORN: COMMODORE 64. Guía del usuario.
HEILBORN: Programas para ciencias e ingeniería. Edición APPLE II.
HEILBORN: VIC 20. Guía del usuario.
HOFFMAN: MSX Guía del usuario.
HOFFMAN: Sistema operativo MS-DOS. Guía del usuario.
HOGAN: Sistema operativo CP/M. Guía del usuario (2.ª ed.)
JEFFRIES: COMMODORE 64. Pasatiempos y juegos.
JOHNSON: CP/M. Manual para programadores.
MOTTOLA: Programación en lenguaje ensamblador para el APPLE II.
OSBORNE: Guía del comprador de sistemas de gestión.
OSBORNE: Guía del ordenador personal PET/CBM.
POOLE: Algunos programas de uso común en BASIC.
POOLE: Algunos programas de uso común en BASIC. Edición APPLE II.
POOLE: Algunos programas de uso común en BASIC. Edición ATARI.
POOLE: Algunos programas de uso común en BASIC. Edición IBM.
POOLE: Algunos programas de uso común en BASIC. Edición PET/CBM.
POOLE: Algunos programas de uso común en BASIC. Edición TRS-80.
POOLE: Algunos programas de uso común en PASCAL.
POOLE: APPLE II. Guía del usuario.
POOLE: Programas prácticos en BASIC.
POOLE: Programas prácticos en BASIC. Edición APPLE II.
POOLE: Programas prácticos en BASIC. Edición IBM.
POOLE: Programas prácticos en BASIC. Edición TRS-80.
POOLE: Programas prácticos en PASCAL.
SACHS: El IBM/PC.
SAND: PASCAL avanzado. Técnicas de programación.
STEWART: Juegos y programas educativos para C-64.
STEWART: Juegos y programas educativos para Apple.
TOWNSEND: Aplique el dBASE.
THOMAS: Sistema operativo UNIX. Guía del usuario.
WAITE: Introducción al procesamiento de palabras.

DISCOGUIAS PUBLICADOS POR OSBORNE/McGRAW-HILL

GIFFORD: Discoguía para APPLE II.
INGRAHAM: Discoguía para CP/M.
JOYANES: Discoguía para ZX SPECTRUM (TS2068).
TAYLOR: Discoguía para ATARI 400/800.
WILSON: Discoguía para IBM/PC.
WILSON: Discoguía para VISICALC.

OTRAS OBRAS DE INTERES PUBLICADAS POR BYTE-BOOKS/McGRAW-HILL

ABELSON: APPLE LOGO.
BOWLES: Introducción al UCSD Pascal.
CIARCIA: Construya una computadora basada en el Z-80 (Guía de diseño y funcionamiento).
DUFF: Introducción al Macintosh.
GABY: GOSUBS 100 subrutinas de uso común para el ZX-81 (TS 1000).
KAMINS: Usted y la microcomputadora (Una introducción humanizada a la microinformática).
KOLVE: Guía para seleccionar y adquirir su microcomputador.
LEWART: Programas de ciencias e ingeniería para microcomputadoras SINCLAIR ZX-81 compatibles con el ZX SPECTRUM.
MORGAN: Introducción al microprocesador 8086/8088 (16 bit).
MULLISH: SINCLAIR ZX-81 (TS 1000). Guía del usuario.
PECKHAM: BASIC para APPLE II. Manual práctico.
PECKHAM: BASIC para COMMODORE 64. Manual práctico.
PECKHAM: BASIC para IBM. Manual práctico.
PECKHAM: BASIC para TRS-80 color. Manual práctico.
SIKONOWIZ: Introducción al IBM/PC.
WATT: Aprendiendo con LOGO.
WATT: Aprendiendo con COMMODORE LOGO.

OTRAS OBRAS DE INTERES PUBLICADAS POR McGRAW-HILL

ADAMIS: Diccionario BASIC.
ADAMIS: Diccionario BASIC del IBM/PC.
ADAMIS: Fórmulas y programas usuales en BASIC.
ADAMIS: Iniciación al BASIC del IBM/PC.
BISHOP: ZX SPECTRUM (TS 2068). Teoría y proyectos de interfase.
GOSLING: Programación estructurada.
HURLEY: Introducción a la programación ZX-81 (TS 1000).
HURLEY: ZX SPECTRUM (TS 2068). Introducción al procesamiento de textos.
PHILLIPS: Programando el DRAGON. Juegos y gráficos.
STREET: ZX SPECTRUM (TS 2068). Técnicas de procesamiento de la información.
WILLIAMS: ZX SPECTRUM (TS 2068). Diseño y programación de juegos.
WOODS: ZX SPECTRUM (TS 2068). Programación en lenguaje ensamblador.

DICCIONARIOS PUBLICADOS POR McGRAW-HILL

FREEDMAN: Glosario de Computación.
SIPPL: Microcomputadoras. Diccionario de Términos.
VAQUERO/JOYANES: INFORMATICA. Glosario de Términos y siglas.

ISBN: 968-451-756-4